aws

Guide de l’utilisateur

Amazon Neptune

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Neptune: Guide de l’utilisateur

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et la présentation commerciale d’Amazon ne peuvent pas être utilisées en relation avec un produit ou un service qui n’appartient pas à Amazon, de toute manière susceptible de créer une confusion chez les clients ou de toute manière dénigrant ou discréditant Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d’Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Présentation de Neptune 1
Dernières mises à jour 4
Commencer à utiliser Neptune 79
Qu'est-ce qu'une base de données orientée graphe ? 79
Pourquoi utiliser des graphes ? 80
Applications de bases de données orientées graphe 81
Langages des requêtes de graphe 85
Exemples de requêtes 85
Cours Neptune en ligne 87
Approfondissement des connaissances 87
Démarrage rapide en utilisant CloudShell 88
Conditions préalables 88
Étape 1 : Connectez-vous à votre cluster Neptune 88
Étape 2 : Insérer et interroger des données 89
Étapes suivantes 90
Utilisation de Neptune avec des carnets graphiques 90
Utilisation du workbench Neptune 92
Déployez avec CloudFormation 96
Activation CloudWatch des journaux 97
Hébergement local 98
Migration vers la version 4 JupyterLab 100
Conditions préalables 104
Configuration du cycle de vie 105
Synchronisation à partir d'un instantané 105
Après la création du bloc-notes 105
Créez le bloc-notes 106
Configuration manuelle des blocs-notes Neptune 107
Magie des carnets Neptune Graph 108
Injection de variable 110
Arguments de requête courants 110
%seed 112
%load 112
%load_ids 112
%load_status 113

%cancel_load ... 113
%status ... 113
%gremlin_status ... 113
%opencypher_status ou %oc_status ... 114
%stream_viewer ... 114
%sparql_status ... 115
%graph_notebook_config ... 115
%graph_notebook_host ... 115
%graph_notebook_version ... 116
%graph_notebook_service ... 116
%graph_notebook_vis_options ... 116
%statistics ... 116
%summary ... 117
%reset_graph ... 117
%get_graph ... 118
% de distribution des diplômes ... 118
%%sparql ... 118
%%gremlin ... 119
%%opencypher ou %%oc ... 120
%%graph_notebook_config ... 121
%%graph_notebook_vis_options ... 122
%neptune_ml ... 122
%%neptune_ml ... 127
Visualisation des données graphiques ... 129
Interface du graphe ... 129
Visualisation Gremlin ... 131
Visualisation SPARQL ... 132
Didacticiels de visualisation ... 133
Résolution des problèmes liés aux ordinateurs portables ... 134
Désynchronisation de l'état du widget ... 134
JupyterLab 4.x retard de l'écran de chargement initial ... 134
Problèmes d'accès à Graph Explorer ... 135
Conseils généraux ... 135
Configuration de Neptune ... 136
Types d'instances de base de données ... 137
Allocation des ressources d'instance ... 137

t3 et t4g 138
Instances r4 139
Instances r5 139
Instances r5d 139
Instances r6g 140
Instances r6i 140
Instances x2g 141
Instances x2iezn 141
Instances x2iedn 141
Instances r8g 141
Instances r7g 142
Instances r7i 143
Instances serverless 144
Types de stockage 144
I/O—Stockage optimisé 145
Créer un cluster Neptune 146
Conditions préalables 147
Créer un cluster 152
Configurer le VPC 155
Ajouter des sous-réseaux 156
Configurer le VPC 157
Création d'un groupe de sécurité 158
DNS dans votre VPC 159
Connexion à un cluster 160
La connectivité réseau 160
Vérifier la connexion 160
Dans le VPC 162
À partir d'un autre VPC 164
Sur un réseau privé 165
Points de terminaison publics 166
Sécurisation de l'accès à Neptune 176
politiques IAM 177
Groupes de sécurité VPC 177
Authentification IAM 177
Accès aux données du graphe 179
Outils 179

Langages de requête 179
Utilisation des outils de ligne de commande 180
Utilisation de Gremlin 184
Utilisation d'openCypher 190
Utilisation de SPARQL 190
Chargement des données 191
Surveillance du cluster Neptune 192
Résolution des problèmes liés au cluster Neptune 192
Neptune sans serveur 194
Cas d'utilisation du mode sans serveur 194
Constaintes 195
Mise à l'échelle de la capacité 196
Définition de la capacité minimale 198
Définition de la capacité maximale 199
Configuration des paramètres de capacité 199
Configuration supplémentaire 201
Configuration mixte 201
Définition des niveaux de promotion 202
Reader-writer alignement 203
Éviter les valeurs de délai d'expiration très élevées 203
Optimisation de la configuration 204
Utilisation du mode sans serveur 204
Création d'un cluster sans serveur 205
Conversion en mode sans serveur 206
Modification de la plage de capacité 207
Conversion en instance provisionnée 207
Configuration du client 207
Contrôle 208
Bases de données mondiales Neptune 210
Présentation de 210
Avantages 212
Limitations 212
Configuration d'une base de données globale 213
Exigences de configuration 214
Création d'une base de données globale 215
Utilisation d'un cluster de bases de données existant comme cluster principal 217

Ajout d'une région secondaire 218
Connexion 219
Gestion d'une base de données globale 220
Suppression d'un cluster 220
Suppression d'une base de données globale 221
Modification d'une base de données globale 221
Reprise après sinistre dans la base de données Neptune 222
Dissociation et promotion 223
Basculement planifié géré 225
Surveillance des déploiements de Neptune 227
Présentation de Neptune 229
Conformité aux normes 230
Conformité avec les normes Gremlin 230
Conformité avec les normes SPARQL 247
Conformité aux spécifications OpenCypher 254
Modèle de données de graphe 268
Dictionnaire 268
Stratégie d'indexation 269
Cache de recherche 273
Cas d'utilisation du cache de recherche 273
Utilisation du cache 274
Sémantique des transactions 276
Niveaux d'isolement 276
Niveaux d'isolement Neptune 277
Exemples de transactions 285
Clusters et instances 290
Instance de base de données principale 290
Read-replica instances 290
Dimensionnement des instances 292
Surveillance des instances 293
Stockage, fiabilité et disponibilité 294
I/O—Stockage optimisé 294
Allocation 295
Facturation du stockage 295
Bonnes pratiques de stockage 296
Fiabilité et haute disponibilité 297

Connexions de point de terminaison 299
Points de terminaison de cluster 299
Points de terminaison du lecteur 300
Points de terminaison d'instance 301
Points de terminaison personnalisés 302
Considérations relatives aux points de terminaison 302
Utilisation des points de terminaison personnalisés 304
Mode Lab 308
Utilisation du mode Lab 308
Index OSGP 310
Dictionnaire de collecte des déchets 310
Sémantique des transactions 310
Support date/heure étendu 311
StrictTimeoutValidation 311
AccurateQRCMemoryEstimation 312
Moteur DFE Neptune 313
Contrôle de l'utilisation du DFE 313
Requêtes exécutées par le DFE 314
Statistiques DFE 316
Limites de taille 317
Statut des statistiques 317
Désactivation du calcul automatique 321
Re-enable calcul automatique 323
Génération manuelle des statistiques 325
Surveillance des statistiques 327
Authentification IAM 329
Suppression des statistiques 332
Erreurs courantes 334
API de résumé de graphe 335
Récupération d'un résumé de graphe 336
Paramètre `mode` 337
Résumé du graphe de propriétés 337
Résumé du graphe RDF 339
Exemple de résumé de graphe de propriétés 340
Exemple de résumé RDF 344
IAM et résumés de graphe 348

Erreurs courantes liées au résumé de graphe 348
Connectivité JDBC 352
Prise en main 352
Utilisation de Tableau 353
Résolution des problèmes 355
Mises à jour du moteur Neptune 357
Exceptions et nouvelles tentatives 358
Pourquoi la logique des nouvelles tentatives est importante 358
Les risques de nouvelles tentatives indiscriminées 358
Élaboration d'une stratégie de réessai pratique 359
Exemples de scénarios 360
Surveillance et observabilité 362
Sécurité 364
Mises à niveau du système d'exploitation Amazon Neptune 365
Minimiser les temps d'arrêt pour les mises à niveau du système d'exploitation nécessitant un redémarrage 365
Appliquer des mises à niveau du système d'exploitation à votre instance de base de données Neptune 366
Protection des données dans les bases de données Neptune 368
Sécurisation des bases de données Neptune avec Amazon VPC 370
Le chiffrement se connecte aux bases de données Neptune 370
Chiffrement des données au repos 372
Authentification avec IAM 376
Les différents rôles 377
Utilisation d'identités 378
Activation d'IAM 379
Connexion avec l'authentification IAM 380
Conditions préalables 382
Connexion depuis la ligne de commande 383
Connexion à partir de la console Garmlin 385
Connexion à l'aide de Gremlin Java 388
Connexion avec SPARQL Java 392
Connexion avec SPARQL et Node.js 395
Connexion avec Python 399
Python Garmlin 403
Javascript 404

Garmlin Go 406
Garmlin.net 411
Utilisation de politiques IAM 413
Identity-Based Politiques 414
Politiques de contrôle des services (SCP) 414
Accès à la console Neptune 415
Association d'une politique 415
Types de politique IAM 416
Utilisation des clés de condition 416
Prise en charge des fonctionnalités IAM 417
Limites des politiques IAM 418
Utilisation des stratégies gérées 418
Utilisation des clés de condition 434
Déclarations de politique administrative de l'IAM 436
Déclarations de politique d'accès aux données personnalisées 461
Service-Linked Rôles 482
Autorisations des rôles 482
Création d'un Service-Linked rôle 484
Modification d'un Service-Linked rôle 485
Supprimer un Service-Linked rôle 485
Informations d'identification temporaires 487
Obtenez des informations d'identification avec le AWS CLI 488
Configuration de Lambda 492
Configuration d'Amazon EC2 493
Journalisation et surveillance 495
Points de terminaison d'un VPC d'interface (AWS PrivateLink) 496
Considérations de conformité 497
Création de déploiements résilients 497
Migration vers Neptune 498
Migration à partir de Neo4j 499
Informations générales 499
Préparation de la migration 503
Provisionnement de l'infrastructure 510
Migrations des données 513
Migration d'une application 520
Compatibilité Neptune 524

Réécritures Cypher 530
Migration des ressources 538
Migration depuis TinkerPop 539
Migration à partir de RDF 540
Utilisation AWS DMS pour migrer 541
Migration à partir de Blazegraph 543
Compatibilité Neptune 543
Provisionnement de l'infrastructure 544
Exportation de données 545
Créer un compartiment Amazon S3 547
Importation des données 548
Chargement des données 550
Chargeur en bloc Neptune 550
Rôle IAM et accès à Amazon S3 552
Formats de données 563
Exemple de chargement 578
Optimisation d'un chargement en bloc 589
Informations de référence sur le chargeur 591
Chargement des données à l'aide de DMS 632
GraphMappingConfig 633
Réplication vers Neptune 637
Charger des données à l'aide de requêtes 643
OpenCypher : neptune.read () 643
SPARQL : CHARGER et DÉCHARGER 643
Gmail : io () étape 644
Stockage 645
Dictionnaire de collecte des déchets 645
ID Edge intégré généré par le serveur 647
Interrogation 650
Mise en file d'attente des requêtes 651
Recherche du nombre de requêtes dans la file d'attente 651
Délais d'expiration des requêtes 651
Cache du plan de requêtes 652
Comment forcer l'activation ou la désactivation du cache du plan de requêtes 652
Comment déterminer si un plan est mis en cache ou non 654
Expulsion 656

Conditions empêchant le plan d'être mis en cache ... 656
ID de requête personnalisé ... 657
Utilisation de l'en-tête HTTP ... 657
Utilisation d'un indicateur de requête SPARQL ... 658
Utilisation de queryId pour vérifier le statut ... 658
Gremlin ... 659
Installation de la console Gremlin ... 662
HTTPS REST ... 668
Java ... 672
Python ... 685
.NET ... 688
Node.js ... 691
Go ... 693
AWS SDK ... 697
Indicateurs de requête ... 701
Statut des requêtes ... 712
Annulation de requêtes ... 717
Sessions basées sur des scripts Gremlin ... 718
Transactions Gremlin ... 721
Résultats en streaming ... 725
Utilisation de l'API Gremlin ... 730
Mise en cache des résultats de requête ... 730
Upserts efficaces à partir de la version 3.6.x ... 738
Upserts efficaces avant la version 3.6.x ... 747
Gremlin `explain` ... 761
Gremlin et DFE ... 815
openCypher ... 818
Comparaison de Gremlin et openCypher ... 819
Utilisation d'openCypher ... 819
Point de terminaison de statut ... 821
Point de terminaison HTTPS ... 828
AWS SDK ... 836
Utilisation du protocole Bolt ... 841
Exemples de requêtes paramétrées ... 866
Modèle de données ... 869
openCypher `explain` ... 870

Transactions 896
Résultats en streaming 905
Indicateurs de requête 911
Restrictions 919
Exceptions 919
Extensions 926
Données spatiales 956
SPARQL 970
Console RDF4J 971
RDF4J Workbench 974
Java 976
API HTTP 980
Indicateurs de requête 994
DESCRIBE et graphe par défaut 1012
Statut des requêtes 1014
Annulation de requêtes 1016
Protocole GSP 1018
SPARQL `explain` 1020
Extension SPARQL `SERVICE` 1053
Outils de visualisation 1056
Graph-explorer 1056
Graph-explorer dans un carnet 1057
Graph-explorer sur Fargate 1057
Démonstration 1060
Logiciel Tom Sawyer 1061
Cambridge Intelligence 1062
Graphistry 1063
metaphacts 1064
G.V() client de base de données de graphes 1065
Fonctionnalités principales 1065
Disponibilité et mises à jour 1067
Démonstration du produit 1067
En savoir plus 1067
Linkurious 1067
Entreprise Linkurious 1068
Ogma 1068

Graph.Build 1070
Graph.Build Atelier : 1070
Graph.Build Transformateurs : 1070
Graph.Build Écrivains : 1071
Exportation de données 1072
neptune-export 1072
Neptune-Export service 1074
Installation du service 1074
Activation de l'accès à Neptune 1078
Permettre l'accès à Neptune-Export 1078
Exécuter une tâche d'exportation 1078
Utilitaire neptune-export 1082
Conditions préalables 1082
Exécution de neptune-export 1084
Exemples de commandes 1084
Fichiers exportés 1085
Paramètres d'exportation 1086
command 1089
outputS3Path 1089
expectedBucketOwner 1089
jobSize 1089
params 1090
additionalParams 1090
params 1092
Exemples de filtrage 1104
Résolution des problèmes 1109
Erreurs courantes 1110
Exportation des résultats de requêtes Gremlin vers Amazon S3 1111
Exemples 1112
Conditions préalables 1112
Format de la réponse 1114
Octroi de l'accès à la fonctionnalité d'exportation Amazon S3 de Gremlin 1117
Gestion de Neptune 1121
Solution Neptune Blue/Green 1123
Prérequis pour Neptune Blue/Green 1124
CloudFormation À utiliser pour exécuter la solution 1125

Surveillance de la progression 1126
Passage au cluster mis à jour 1129
Nettoyage 1130
Bonnes pratiques 1130
Résolution des problèmes 1131
Autorisations des utilisateurs IAM 1132
Politique de rôle liée à un service 1132
Création d'un utilisateur IAM 1133
Groupes de paramètres 1135
Modification d'un groupe de paramètres 1138
Création d'un groupe de paramètres 1138
Parameters 1140
neptune_enable_audit_log 1141
neptune_enable_slow_query_log 1141
neptune_slow_query_log_threshold 1141
neptune_lab_mode 1142
neptune_query_timeout 1142
neptune_streams 1143
neptune_streams_expiry_days 1143
neptune_lookup_cache 1144
neptune_autoscaling_config 1144
neptune_ml_iam_role 1145
neptune_ml_endpoint 1145
neptune_enable_inline_server_generated_edge_id 1145
neptune_dfe_query_engine 1146
neptune_query_timeout 1146
neptune_result_cache 1147
UndoLogPurgeConfig 1147
neptune_enforce_ssl 1147
Lancement à l'aide de la console 1148
Arrêt et démarrage d'un cluster 1156
Présentation de l'arrêt et du démarrage 1156
Arrêt d'un cluster 1157
Démarrage d'un cluster de bases de données 1158
API de réinitialisation rapide 1160
Utilisation de l'authentification IAM 1163

Magie %db_reset 1164
Erreurs courantes 1165
Ajout d'instances de lecteur 1167
Création d'une instance de lecteur 1168
Modification d'un cluster de base de données 1170
Modifier une instance 1171
Performances et dimensionnement 1173
Dimensionnement du stockage 1173
Mise à l'échelle d'une instance ; 1173
Dimensionnement en lecture 1173
Scalabilité automatique 1175
Autoscaling et mode sans serveur 1177
Activation de l'autoscaling 1178
Suppression de l'autoscaling 1181
Maintenance du cluster 1182
Numéros de version 1182
Types de version 1183
Durée de vie des versions du moteur 1186
Gestion des mises à jour du moteur 1187
Processus de mise à niveau 1193
Mise à niveau vers la version 1.2.0.0 ou supérieure 1195
Mettre à jour via CloudFormation 1197
1.2.0.1 vers 1.2.0.2 1198
1.1.1.0 vers 1.2.0.2, par défaut 1201
1.1.1.0 vers 1.2.0.2, personnalisée 1203
1.1.1.0 vers 1.2.0.2, mixte 1206
Clonage d'un cluster de base de données 1210
Limites 1212
Copy-on-Write Protocole 1212
Suppression d'une base de données source 1215
Gestion des instances 1216
Instances extensibles T3 1217
Modification d'une instance 1220
Changement de nom d'une instance de base de données Neptune 1226
Redémarrage d'une instance DB 1228
Suppression d'une instance de base de données 1230

Flux Neptune 1233
Utilisation de Streams 1236
Activation de Streams 1236
Désactivation de Streams 1237
Appel de l'API Streams 1237
Réponse Streams 1239
Lecture des données du flux 1242
Exceptions Streams 1244
Formats d'enregistrements Streams 1245
PG_JSON 1246
RDF-NQUADS 1249
Exemples d'utilisation de Streams 1249
Exemples AT_SEQUENCE_NUMBER 1250
Exemple AFTER_SEQUENCE_NUMBER 1251
Exemple TRIM_HORIZON 1252
Exemple LATEST 1252
Exemple Compression 1254
Neptune-to-Neptune Configuration de la réplication 1255
Choisissez un CloudFormation modèle 1255
Ajout de détails sur la pile 1257
Exécuter le modèle 1261
Mise à jour de l'interrogateur de flux 1262
Utilisation des flux pour la reprise après sinistre 1263
Configuration de la réplication 1264
Autres considérations 1268
Recherche en texte intégral Neptune 1269
Full-text configuration de la recherche 1271
CloudFormation modèle 1272
Bases de données existantes 1279
Mise à jour de l'interrogateur 1281
Arrêt et démarrage de l'interrogateur de flux 1282
OpenSearch Sans serveur 1283
Modifications requises pour la fonction AWS Lambda du poller 1283
Modifications requises pour l'utilitaire d'importation en bloc 1283
Modifications nécessaires au rôle IAM utilisé pour interroger Neptune 1284
Interrogation avec le contrôle d'accès détaillé 1285

Utilisation de la syntaxe Lucene 1285
Modèle de données de recherche en texte intégral Neptune 1286
Exemple de document SPARQL 1288
Exemple de document Gremlin 1289
Paramètres de recherche en texte intégral 1290
Indexation hors chaîne 1295
Mise à jour d'une pile existante 1296
Exclusion de champs 1297
Mappage des types de données 1300
Validation des types de données 1302
Exemples de requêtes 1308
Full-text-search exécution de requêtes 1310
Exemples de requêtes de recherche en texte intégral SPARQL 1312
Requête de correspondance 1312
prefix 1312
fuzzy 1313
term 1313
query_string 1313
simple_query_string 1314
tri par champ de chaîne 1314
tri par champ hors chaîne 1314
tri par ID 1315
tri par étiquette 1315
tri par doc_type 1316
Syntaxe de Lucene 1316
Exemple de requêtes de recherche en texte intégral Gremlin 1317
Requête de base `match` 1318
`match` 1318
`fuzzy` 1318
Requête approximative `query_string` 1318
Expression régulière `query_string` 1319
Requête hybride 1319
Exemple de recherche en texte intégral 1319
`query_string`, '+' et '-' 1320
`query_string`, `AND` et `OR` 1321
`term` 1322

`prefix` ... 1322
Syntaxe de Lucene ... 1322
TinkerPop Graphe moderne ... 1324
Tri par champ de chaîne ... 1324
Tri par champ hors chaîne ... 1325
Tri par champ d'ID ... 1325
Tri par champ d'étiquette ... 1325
Tri par champ `document_type` ... 1325
Dépannage et métriques ... 1326
Dépannage des lectures ... 1327
Dépannage des écritures ... 1327
Out-of-sync problèmes ... 1328
AWS Lambda fonctions ... 1330
Connexions au Kremlin WebSocket ... 1330
Recommandations sur Gremlin Lambda ... 1331
Recommandations relatives aux demandes d'écriture ... 1332
Recommandations relatives aux demandes de lecture ... 1333
Latence des démarrages à froid ... 1334
Création d'une fonction lambda ... 1334
Exemples de fonctions Lambda ... 1338
Exemple Java ... 1338
JavaScript exemple ... 1344
Exemple Python ... 1348
Machine learning Neptune ... 1353
Fonctionnalités de Neptune ML ... 1353
Configuration de Neptune ML ... 1356
Configuration à l'aide de CloudFormation ... 1357
Configuration manuelle ... 1361
En utilisant le AWS CLI ... 1370
Utilisation de Neptune ML ... 1374
Flux de travail initial ... 1374
Gestion de l'évolution des données ... 1377
Mise à jour des artefacts de modèle ... 1378
Flux de travail de modèle personnalisé ... 1379
Sélection d'instances ... 1380
Pour le traitement des données ... 1380

Pour l'entraînement de modèle et la transformation de modèle 1380
Appeler un point de terminaison d'inférence 1381
Exportation de données 1382
Exemples d'exportation Neptune 1382
Réglages des paramètres 1385
AdditionalParams 1386
targets 1389
fonctionnalités 1396
Exemples 1406
Traitement des données 1422
Gestion du traitement de données 1422
Traitement mis à jour 1424
Encodage des fonctionnalités 1427
Modification d'un fichier de données d'entraînement 1436
Entraînement de modèle 1448
Modèles et entraînement 1455
Personnalisation des hyperparamètres 1459
Bonnes pratiques d'entraînement 1472
Transformation de modèle 1476
Inférence incrémentielle 1476
Transformation de modèle pour n'importe quelle tâche 1478
Artefacts de modèle 1481
Artefacts pour différentes tâches 1481
Génération de nouveaux artefacts 1481
Modèles personnalisés 1485
Vue d'ensemble des modèles personnalisés 1486
Développement de modèle personnalisé 1493
Point de terminaison d'inférence 1499
Gestion des points de terminaison d'inférence 1499
Requêtes d'inférence 1503
Requêtes d'inférence Gremlin 1504
Requêtes d'inférence SPARQL 1529
API Neptune ML 1536
Commande dataprocessing 1538
Commande modeltraining 1550
Commande modeltransform 1566

Commande endpoints 1580
Exceptions 1594
Limites 1595
SageMaker Limites de l'IA 1596
Surveillance de Neptune 1597
Statut d'une instance 1598
Exemple de sortie. 1601
En utilisant CloudWatch 1602
Utilisation de la console 1603
À l'aide du AWS CLI 1603
Utilisation de l' CloudWatch API 1604
Surveillance des performances d'instance 1605
Métriques Neptune 1606
Dimensions Neptune 1640
Journaux d'audit avec Neptune 1641
Activation des journaux d'audit 1641
Consultation des journaux d'audit 1641
Détails du journal d'audit 1642
Logs Neptune CloudWatch 1643
Publier des journaux dans CloudWatch des journaux (console) 1644
Publier les journaux d'audit dans CloudWatch Logs (CLI) 1644
Publier les journaux à requêtes lentes dans Logs (CLI) CloudWatch 1645
Surveillance des événements de journaux 1645
CloudWatch Journaux pour ordinateurs portables 1646
Slow-query journaux 1648
Affichage des journaux dans la console 1649
Slow-query fichiers journaux 1649
Attributs en mode `info` 1650
Attributs en mode `debug` 1653
Exemple de sortie 1655
Journalisation des appels d'API Neptune avec AWS CloudTrail 1657
Informations sur Neptune dans CloudTrail 1658
Présentation des entrées des fichiers journaux 1659
Événements 1661
Catégories et messages 1662
Affichage d'événements 1679

Abonnement aux événements 1681
Gestion des abonnements 1681
Marquage des ressources Neptune 1682
Présentation du balisage 1683
Marquage dans la console 1686
Marquage à l'aide de la CLI 1687
Balisage avec l'API 1687
Utilisation des ARN 1689
Sauvegarde et restauration 1694
Sauvegardes 1694
Tolérance aux pannes 1695
Métriques de sauvegarde 1697
Restauration des données 1698
Fenêtre de sauvegarde 1699
Création d'un instantané 1700
Utilisation de la console 1700
Restaurer à partir d'un instantané 1701
Considérations importantes relatives à la restauration 1701
Restauration en cours 1703
Copie d'un instantané 1705
Limitations 1705
Conservation des copies d'instantané 1706
Chiffrement 1706
Cross-Region Copie d'instantanés 1707
Copie d'un instantané sur la console 1707
Copier un instantané à l'aide du AWS CLI 1709
Partage d'un instantané 1712
Instantanés chiffrés 1713
Partage 1716
Suppression d'un instantané 1719
Utilisation de la console 1719
À l'aide du AWS CLI 1719
Utilisation de l'API Neptune 1719
Bonnes pratiques 1720
Directives opérationnelles de base 1722
Sécurité 1724

Éviter différentes tailles d'instance 1725
Éviter les redémarrages pendant le chargement en bloc 1725
Si vous avez de nombreux prédicats 1725
Éviter les transactions de longue durée 1726
Utilisation de métriques 1726
Réglage des requêtes 1727
Équilibrage de charge 1728
Utilisation d'une instance temporaire 1728
Redimensionnement d'une instance 1729
Erreur d'interruption de tâche 1729
Gremlin (général) 1730
Configuration du rythme cardiaque 1731
Optimisation des requêtes upsert 1733
Différences d'exécution 1733
Écritures multithreads 1736
Élagage d'enregistrements 1737
`datetime( )` 1738
Date et heure natives 1738
Gremlin (client Java) 1741
Réutilisation de l'objet client 1741
Clients distincts pour la lecture et l'écriture 1742
Points de terminaison de réplica multiples 1742
Fermeture du client en fin de session 1743
Nouvelle connexion après basculement 1743
Ensemble maxInProcess PerConnection = maxSimultaneousUsage PerConnection 1743
Envoi de requêtes sous la forme de bytecode 1744
Utilisation intégrale des résultats de requête 1745
Ajout en bloc de sommets et d'arêtes 1746
Désactivation de la mise en cache du DNS sur la machine virtuelle Java 1747
Délais d'expiration au niveau de chaque requête 1747
Gestion d'une exception `TimeoutException` 1749
openCypher et Bolt 1750
Reconnexion après basculement 1751
Réutilisation de l'objet Driver 1751
Gestion des connexions Lambda 1752
Préférer les arêtes dirigées 1752

Aucune requête de transaction simultanée 1753
Fermer les objets Driver 1753
Utilisation des modes de transaction explicites 1754
Logique des nouvelles tentatives 1757
Utilisation de la clause SET 1760
Utilisation des requêtes paramétrées 1761
UNWIND classe 1762
Expressions de chemin de longueur variable (VLP) 1763
Utiliser des noms de relations granulaires 1764
Spécifier les étiquettes des bords 1765
Évitez la clause WITH 1766
Filtres restrictifs 1767
Vérifier les propriétés de manière explicite 1767
Évitez le chemin nommé 1768
Évitez COLLECT (DISTINCT ()) 1769
Préférez la fonction des propriétés 1769
Calculs statiques 1770
Entrées par lots 1770
Préférez la personnalisation IDs 1771
Évitez les calculs de ~id 1772
Mise à jour/fusion de plusieurs nœuds 1773
SPARQL 1774
Interroger tous les graphes nommés 1774
Spécifier un graphe nommé à charger 1775
FILTER ET VALUES 1775
Amazon ECS et Amazon EKS 1777
Cycle de vie des connexions 1778
Remplacement de l'hôte 1779
Réseautage et IAM 1780
AWS CLI et SDK 1783
Limites Neptune 1786
Régions 1786
Régions chinoises 1787
Asie-Pacifique (Hyderabad) 1787
Taille du volume du cluster 1787
Tailles d'instance 1788

Par compte 1788
VPC obligatoire 1788
SSL obligatoire 1788
Zones de disponibilité et groupes de sous-réseaux 1789
Charge utile des demandes HTTP 1789
Gremlin 1789
Pas de caractères nuls 1789
SPARQL UPDATE LOAD 1790
Authentification et accès 1790
WebSockets Limites 1791
Propriétés et étiquettes 1794
Bulk-loading 1794
Intégrations Neptune 1796
Outils et utilitaires 1798
Utilitaire GraphQL 1798
Installation et configuration 1800
Utilisation des données existantes 1800
Utilisation d'un schéma sans directives 1801
Utilisation des directives 1807
Command-line arguments 1812
Flux de nœuds 1816
Erreurs Neptune 1818
Instances de base de données disponibles insuffisantes 1818
Codes d'erreur du moteur 1819
Format d'erreur 1819
Erreurs liées aux requêtes 1819
Erreurs d'IAM 1825
Erreurs d'API 1827
Erreurs du chargeur 1830
Versions du moteur 1833
Planification de la durée de vie des versions du moteur 1836
Sortie : 1.4.7.0 (2026-03-03) 1837
Caractéristiques 1837
Améliorations 1838
Défauts corrigés 1838
Query-Language Versions prises en charge 1839

Chemins de mise à niveau 1839
Mise à niveau 1839
Version : 1.4.6.3 (2025-12-18) 1841
Défauts corrigés 1842
Versions de langage de requête prises en charge 1842
Chemins de mise à niveau 1843
Mise à niveau 1843
Version : 1.4.6.2 (2025-11-18) 1845
Défauts corrigés 1845
Versions de langage de requête prises en charge 1846
Chemins de mise à niveau 1846
Mise à niveau 1846
Version : 1.4.6.1 (2025-09-18) 1849
Défauts corrigés 1849
Versions de langage de requête prises en charge 1849
Chemins de mise à niveau 1849
Mise à niveau 1850
Version : 1.4.6.0 (2025-09-02) 1852
Nouvelles fonctionnalités 1852
Améliorations 1853
Défauts corrigés 1853
Versions de langage de requête prises en charge 1854
Chemins de mise à niveau 1854
Mise à niveau 1854
Version : 1.4.5.1 (2025-06-30) 1857
Défauts corrigés 1857
Versions de langage de requête prises en charge 1857
Chemins de mise à niveau 1857
Mise à niveau 1857
Version : 1.4.5.0 (2025-04-09) 1860
Nouvelles fonctionnalités 1860
Améliorations 1861
Défauts corrigés 1862
Versions de langage de requête prises en charge 1863
Chemins de mise à niveau 1863
Mise à niveau 1863

Version : 1.4.4.0 (2025-02-24) 1866
Défauts corrigés 1867
Versions de langage de requête prises en charge 1867
Chemins de mise à niveau 1867
Mise à niveau 1867
Version : 1.4.3.0 (2025-01-21) 1870
Nouvelles fonctionnalités 1870
Améliorations 1871
Défauts corrigés 1872
Versions de langage de requête prises en charge 1872
Chemins de mise à niveau 1872
Mise à niveau 1873
Version : 1.4.2.0 (2024-12-19) 1875
Défauts corrigés 1876
Versions de langage de requête prises en charge 1876
Chemins de mise à niveau 1876
Mise à niveau 1876
Version : 1.4.1.0 (2024-11-21) 1879
Nouvelles fonctionnalités 1879
Améliorations 1880
Défauts corrigés 1881
Versions de langage de requête prises en charge 1882
Chemins de mise à niveau 1882
Mise à niveau 1882
Version : 1.4.0.0 (2024-11-06) 1884
Nouvelles fonctionnalités 1885
Améliorations 1886
Défauts corrigés 1887
Versions de langage de requête prises en charge 1888
Chemins de mise à niveau 1888
Mise à niveau 1888
Version : 1.3.4.0 (2024-10-01) 1891
Améliorations 1892
Défauts corrigés 1892
Versions de langage de requête prises en charge 1894
Chemins de mise à niveau 1894

Mise à niveau 1894
Version : 1.3.3.0 (2024-08-05) 1896
Défauts corrigés 1897
Atténuation des problèmes dans la version 1.3.3.0 1898
Versions de langage de requête prises en charge 1899
Chemins de mise à niveau 1899
Mise à niveau 1899
Version : 1.3.2.1 (2024-06-20) 1901
Défauts corrigés 1902
Les modifications apportées au 1.3.2.1 sont reportées du 1.3.2.0 1904
Chemins de mise à niveau 1909
Mise à niveau 1909
Version : 1.3.2.0 (2024-06-10) 1911
Améliorations 1912
Défauts corrigés 1914
Atténuation des problèmes dans la version 1.3.2.0 1916
Versions de langage de requête prises en charge 1918
Chemins de mise à niveau 1918
Mise à niveau 1918
Sortie : 1.3.1.0 (2024-03-06) 1921
Améliorations 1921
Défauts corrigés 1922
Versions de langage de requête prises en charge 1923
Chemins de mise à niveau 1923
Mise à niveau 1923
Version : 1.3.0.0 (15-11-2023) 1925
Nouvelles fonctions 1926
Améliorations 1926
Défauts corrigés 1928
Versions de langage de requête prises en charge 1930
Chemins de mise à niveau 1930
Mise à niveau 1930
Version : 1.2.1.2 (2024-08-05) 1932
Défauts corrigés 1934
Versions de langage de requête prises en charge 1934
Chemins de mise à niveau 1934

Mise à niveau 1934
Sortie : 1.2.1.1 (2024-03-11) 1937
Améliorations 1938
Défauts corrigés 1938
Versions de langage de requête prises en charge 1939
Chemins de mise à niveau 1940
Mise à niveau 1940
Sortie : 1.2.1.0 (08/03/2023) 1942
Versions de correctifs 1943
Nouvelles fonctions 1944
Améliorations 1945
Défauts corrigés 1946
Versions de langage de requête prises en charge 1947
Chemins de mise à niveau 1948
Mise à niveau 1948
Version : 1.2.1.0.R7 (06-10-2023) 1950
Sortie : 1.2.1.0.R6 (12/09/2023) 1954
Sortie : 1.2.1.0.R5 (02/09/2023) 1958
Sortie : 1.2.1.0.R4 (10/08/2023) 1962
Sortie : 1.2.1.0.R3 (13/06/2023) 1967
Sortie : 1.2.1.0.R2 (02/05/2023) 1973
Sortie : 1.2.0.2 (20/11/2022) 1977
Versions de correctifs 1978
Nouvelles fonctions 1979
Améliorations 1979
Versions de langage de requête prises en charge 1979
Chemins de mise à niveau 1980
Mise à niveau 1980
Sortie : 1.2.0.2.R6 (12/09/2023) 1982
Sortie : 1.2.0.2.R5 (16/08/2023) 1986
Sortie : 1.2.0.2.R4 (08/05/2023) 1990
Sortie : 1.2.0.2.R3 (27/03/2023) 1994
Sortie : 1.2.0.2.R2 (15/12/2022) 1999
Sortie : 1.2.0.1 (26/10/2022) 2003
Versions de correctifs 2004
Nouvelles fonctions 2005

Améliorations 2005
Défauts corrigés 2005
Versions de langage de requête prises en charge 2006
Chemins de mise à niveau 2006
Mise à niveau 2006
Version de maintenance : 1.2.0.1.R3 (27/09/2023) 2008
Version de maintenance : 1.2.0.1.R2 (13/12/2022) 2013
Sortie : 1.2.0.0 (21/07/2022) 2017
Versions de correctifs 2018
Nouvelles fonctions 2019
Améliorations 2019
Défauts corrigés 2021
Versions de langage de requête prises en charge 2022
Chemins de mise à niveau 2023
Mise à niveau 2023
Sortie : 1.2.0.0.R4 (29/09/2023) 2026
Sortie : 1.2.0.0.R3 (15/12/2022) 2031
Sortie : 1.2.0.0.R2 (14/10/2022) 2036
Sortie : 1.1.1.0 (19/04/2022) 2042
Versions de correctifs 2043
Nouvelles fonctions 2044
Améliorations 2045
Défauts corrigés 2046
Versions de langage de requête prises en charge 2047
Chemins de mise à niveau 2047
Mise à niveau 2047
Sortie : 1.1.1.0.R7 (23/01/2023) 2051
Sortie : 1.1.1.0.R6 (23/09/2022) 2056
Sortie : 1.1.1.0.R5 (21/07/2022) 2062
Sortie : 1.1.1.0.R4 (23/06/2022) 2067
Sortie : 1.1.1.0.R3 (07/06/2022) 2072
Version de maintenance 1.1.1.0.R2 (16/05/2022) 2078
Sortie : 1.1.0.0 (19/11/2021) 2083
Versions de correctifs 2084
Nouvelles fonctions 2084
Améliorations 2085

Défauts corrigés 2087
Versions de langage de requête prises en charge 2087
Chemins de mise à niveau 2087
Mise à niveau 2087
Version de maintenance 1.1.0.0.R3 (23/12/2022) 2090
Version de maintenance 1.1.0.0.R2 (16/05/2022) 2095
Sortie : 1.0.5.1 (01/10/2021) 2100
Versions de correctifs 2100
Nouvelles fonctions 2100
Améliorations 2100
Défauts corrigés 2101
Versions de langage de requête prises en charge 2101
Chemins de mise à niveau 2101
Mise à niveau 2102
Version de maintenance 1.0.5.1.R4 (16/05/2022) 2104
Sortie : 1.0.5.1.R3 (13/01/2022) 2106
Sortie : 1.0.5.1.R2 (26/10/2021) 2109
Sortie : 1.0.5.0 (27/07/2021) 2112
Versions de correctifs 2112
Nouvelles fonctions 2112
Améliorations 2113
Défauts corrigés 2114
Versions de langage de requête prises en charge 2114
Chemins de mise à niveau 2115
Mise à niveau 2115
Version de maintenance 1.0.5.0.R5 (16/05/2022) 2117
Sortie : 1.0.5.0.R3 (15/09/2021) 2120
Sortie : 1.0.5.0.R2 (16/08/2021) 2123
Sortie : 1.0.4.2 (01/06/2021) 2126
Sortie : 1.0.4.2.R5 (16/08/2021) 2126
Sortie : 1.0.4.2.R4 (23/07/2021) 2127
Sortie : 1.0.4.2.R3 (28/06/2021) 2128
Sortie : 1.0.4.2.R2 (01/06/2021) 2129
Sortie : 1.0.4.2.R1 (27/05/2021) 2133
Sortie : 1.0.4.1 (08/12/2020) 2133
Versions de correctifs 2133

Nouvelles fonctions 2134
Améliorations 2134
Défauts corrigés 2135
Versions de langage de requête prises en charge 2135
Chemins de mise à niveau 2136
Mise à niveau 2136
Sortie : 1.0.4.1.R1.1 (22/03/2021) 2138
Sortie : 1.0.4.1.R2 (24/02/2021) 2141
Sortie : 1.0.4.0 (12/10/2020) 2147
Versions de correctifs 2147
Nouvelles fonctions 2147
Améliorations 2147
Défauts corrigés 2149
Versions de langage de requête prises en charge 2149
Chemins de mise à niveau 2149
Mise à niveau 2149
Sortie : 1.0.4.0.R2 (24/02/2021) 2152
Sortie : 1.0.3.0 (03/08/2020) 2155
Versions de correctifs 2155
Nouvelles fonctions 2155
Améliorations 2156
Défauts corrigés 2156
Versions de langage de requête prises en charge 2157
Chemins de mise à niveau 2157
Mise à niveau 2157
Sortie : 1.0.3.0.R3 (19/02/2021) 2159
Sortie : 1.0.3.0.R2 (12/10/2020) 2163
Sortie : 1.0.2.2 (09/03/2020) 2166
Versions de correctifs 2166
Améliorations 2167
Défauts corrigés 2167
Versions de langage de requête prises en charge 2168
Chemins de mise à niveau 2168
Mise à niveau 2168
Sortie : 1.0.2.2.R6 (19/02/2021) 2170
Sortie : 1.0.2.2.R5 (12/10/2020) 2173

Sortie : 1.0.2.2.R4 (23/07/2020) 2177
Sortie : 1.0.2.2.R3 (22/07/2020) 2180
Sortie : 1.0.2.2.R2 (02/04/2020) 2180
Sortie : 1.0.2.1 (22/11/2019) 2183
Versions de correctifs 2183
Nouvelles fonctions 2183
Améliorations 2184
Défauts corrigés 2184
Versions de langage de requête prises en charge 2185
Chemins de mise à niveau 2185
Mise à niveau 2185
Sortie : 1.0.2.1.R6 (22/04/2020) 2187
Sortie : 1.0.2.1.R5 (22/04/2020) 2191
Sortie : 1.0.2.1.R4 (20/12/2019) 2191
Sortie : 1.0.2.1.R3 (12/12/2019) 2194
Sortie : 1.0.2.1.R2 (25/11/2019) 2197
Sortie : 1.0.2.0 (08/11/2019) 2199
IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE 2199
Versions de correctifs 2200
Nouvelles fonctions 2200
Versions de langage de requête prises en charge 2200
Chemins de mise à niveau 2200
Mise à niveau 2201
Sortie : 1.0.2.0.R3 (05/05/2020) 2203
Sortie : 1.0.2.0.R2 (21/11/2019) 2206
Sortie : 1.0.1.2 (10/06/2020) 2209
IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE 2209
Améliorations 2209
Défauts corrigés 2210
Versions de langage de requête prises en charge 2210
Sortie : 1.0.1.1 (26/06/2020) 2210
IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE 2210
Défauts corrigés 2210
Versions de langage de requête prises en charge 2211
Sortie : 1.0.1.0 (02/07/2019) 2211
IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE 2211

Version 1.0.1.0.200502.0 (31/10/2019) 2211
Version 1.0.1.0.200463.0 (15/10/2019) 2212
Sortie : 1.0.1.0.200457.0 (19/09/2019) 2213
Sortie : 1.0.1.0.200369.0 (13/08/2019) 2214
Sortie : 1.0.1.0.200366.0 (26/07/2019) 2215
Sortie : 1.0.1.0.200348.0 (02/07/2019) 2217
Versions antérieures 2218
Utiliser Neptune APIs 2232
Actions IAM partagées 2232
Référence de l'API de gestion 2239
Clusters 2246
Créez DBCluster 2246
Supprimer DBCluster 2259
Modifier DBCluster 2266
Démarrer DBCluster 2277
Arrêtez DBCluster 2284
AddRoleToDBCluster 2290
RemoveRoleFromDBCluster 2291
Basculement DBCluster 2292
PromoteReadReplicaDBCluster 2299
Décrivez DBClusters 2305
_____ 2307
DBCluster 2307
DBClusterMembre 2314
DBClusterRôle 2314
CloudwatchLogsExportConfiguration 2315
PendingCloudwatchLogsExports 2316
ClusterPendingModifiedValues 2316
Bases de données globales 2317
CreateGlobalCluster 2318
DeleteGlobalCluster 2321
ModifyGlobalCluster 2322
DescribeGlobalClusters 2325
FailoverGlobalCluster 2327
RemoveFromGlobalCluster 2329
_____ 2331

GlobalCluster 2331
GlobalClusterMember 2333
instances 2333
Créez DBInstance 2334
Supprimer DBInstance 2347
Modifier DBInstance 2354
Redémarrer DBInstance 2367
Décrivez DBInstances 2373
DescribeOrderableDBInstanceOptions 2375
DescribeValidDBInstanceModifications 2376
_____ 2377
DBInstance 2377
DBInstanceStatusInfo 2383
Option commandable DBInstance 2383
PendingModifiedValues 2385
ValidStorageOptions 2387
Valide DBInstance ModificationsMessage 2388
Paramètres 2388
DBParameterGroupe de copies 2389
Copier DBCluster ParameterGroup 2391
Créer un DBParameter groupe 2393
Créez DBCluster ParameterGroup 2395
Supprimer le DBParameter groupe 2397
Supprimer DBCluster ParameterGroup 2398
Modifier le DBParameter groupe 2399
Modifier DBCluster ParameterGroup 2401
Réinitialiser le DBParameter groupe 2402
Réinitialiser DBCluster ParameterGroup 2404
Décrivez DBParameters 2405
Décrire DBParameter les groupes 2407
Décrire DBCluster les paramètres 2408
Décrivez DBCluster ParameterGroups 2409
DescribeEngineDefaultParameters 2411
DescribeEngineDefaultClusterParameters 2412
_____ 2413
Paramètre 2413

DBParameterGroupe 2414
DBClusterParameterGroup 2415
DBParameterGroupStatus 2416
Sous-réseaux 2416
Créer un DBSubnet groupe 2417
Supprimer le DBSubnet groupe 2419
Modifier le DBSubnet groupe 2420
Décrire DBSubnet les groupes 2421
_____ 2423
Sous-réseau 2423
DBSubnetGroupe 2423
Instantanés 2424
Créer un DBCluster instantané 2425
Supprimer DBCluster un instantané 2428
Copier DBCluster un instantané 2431
Modifier DBCluster SnapshotAttribute 2436
Restaurer DBCluster FromSnapshot 2438
Restaurer DBCluster ToPointInTime 2449
Décrire les DBCluster instantanés 2459
Décrivez DBCluster SnapshotAttributes 2462
_____ 2464
DBClusterInstantané 2464
DBClusterSnapshotAttribute 2466
DBClusterSnapshotAttributesResult 2467
Événements 2468
CreateEventSubscription 2468
DeleteEventSubscription 2472
ModifyEventSubscription 2473
DescribeEventSubscriptions 2476
AddSourceIdentifierToSubscription 2477
RemoveSourceIdentifierFromSubscription 2479
DescribeEvents 2481
DescribeEventCategories 2483
_____ 2484
Événement 2484
EventCategoriesMap 2485

EventSubscription ... 2485
Autre ... 2487
AddTagsToResource ... 2487
ListTagsForResource ... 2488
RemoveTagsFromResource ... 2489
ApplyPendingMaintenanceAction ... 2490
DescribePendingMaintenanceActions ... 2491
Décrire DBEngine les versions ... 2492
_____ ... 2494
DBEngineVersion ... 2494
EngineDefaults ... 2496
PendingMaintenanceAction ... 2496
ResourcePendingMaintenanceActions ... 2497
UpgradeTarget ... 2498
Tag ... 2499
Types de données ... 2499
AvailabilityZone ... 2500
DBSecurityGroupMembership ... 2500
DomainMembership ... 2500
DoubleRange ... 2501
Point de terminaison ... 2501
Filtre ... 2502
Range ... 2502
Serverless V2 ScalingConfiguration ... 2502
Serverless V2 ScalingConfigurationInfo ... 2503
Fuseau horaire ... 2504
VpcSecurityGroupMembership ... 2504
Anomalies d'API ... 2504
AuthorizationAlreadyExistsFault ... 2507
AuthorizationNotFoundFault ... 2507
AuthorizationQuotaExceededFault ... 2507
CertificateNotFoundFault ... 2508
DBClusterAlreadyExistsFault ... 2508
DBClusterNotFoundFault ... 2508
DBClusterParameterGroupNotFoundFault ... 2509
DBClusterQuotaExceededFault ... 2509

DBClusterRoleAlreadyExistsFault 2509
DBClusterRoleNotFoundFault 2510
DBClusterRoleQuotaExceededFault 2510
DBClusterSnapshotAlreadyExistsFault 2510
DBClusterSnapshotNotFoundFault 2511
DBInstanceAlreadyExistsFault 2511
DBInstanceNotFoundFault 2511
DBLogFileNotFoundFault 2511
DBParameterGroupAlreadyExistsFault 2512
DBParameterGroupNotFoundFault 2512
DBParameterGroupQuotaExceededFault 2512
DBSecurityGroupAlreadyExistsFault 2513
DBSecurityGroupNotFoundFault 2513
DBSecurityGroupNotSupportedFault 2513
DBSecurityGroupQuotaExceededFault 2514
DBSnapshotAlreadyExistsFault 2514
DBSnapshotNotFoundFault 2514
DBSubnetGroupAlreadyExistsFault 2515
DBSubnetGroupDoesNotCoverEnoughAZs 2515
DBSubnetGroupNotAllowedFault 2515
DBSubnetGroupNotFoundFault 2515
DBSubnetGroupQuotaExceededFault 2516
DBSubnetQuotaExceededFault 2516
DBUpgradeDependencyFailureFault 2516
DomainNotFoundFault 2517
EventSubscriptionQuotaExceededFault 2517
GlobalClusterAlreadyExistsFault 2517
GlobalClusterNotFoundFault 2518
GlobalClusterQuotaExceededFault 2518
InstanceQuotaExceededFault 2518
Insuffisant DBCluster CapacityFault 2519
Insuffisant DBInstance CapacityFault 2519
InsufficientStorageClusterCapacityFault 2519
Non valide DBCluster EndpointStateFault 2520
Non valide DBCluster SnapshotStateFault 2520
Non valide DBCluster StateFault 2520

Non valide DBInstance StateFault 2520
Non valide DBParameter GroupStateFault 2521
Non valide DBSecurity GroupStateFault 2521
Non valide DBSnapshot StateFault 2521
Non valide DBSubnet GroupFault 2522
Non valide DBSubnet GroupStateFault 2522
Non valide DBSubnet StateFault 2522
InvalidEventSubscriptionStateFault 2523
InvalidGlobalClusterStateFault 2523
InvalidOptionGroupStateFault 2523
InvalidRestoreFault 2524
InvalidSubnet 2524
Non valide VPCNetwork StateFault 2524
KMSKeyNotAccessibleFault 2524
OptionGroupNotFoundFault 2525
PointInTimeRestoreNotEnabledFault 2525
ProvisionedIopsNotAvailableInAZFault 2525
ResourceNotFoundFault 2526
SNSInvalidTopicFault 2526
SNSNoAuthorizationFault 2526
SNSTopicArnNotFoundFault 2527
SharedSnapshotQuotaExceededFault 2527
SnapshotQuotaExceededFault 2527
SourceNotFoundFault 2527
StorageQuotaExceededFault 2528
StorageTypeNotSupportedFault 2528
SubnetAlreadyInUse 2528
SubscriptionAlreadyExistFault 2529
SubscriptionCategoryNotFoundFault 2529
SubscriptionNotFoundFault 2529
Référence de l'API de données 2530
Général 2534
GetEngineStatus 2534
ExecuteFastReset 2537
_____ 2538
QueryLanguageVersion 2538

FastResetToken 2539
Interrogation 2539
ExecuteGremlinQuery 2540
ExecuteGremlinExplainQuery 2542
ExecuteGremlinProfileQuery 2544
ListGremlinQueries 2546
GetGremlinQueryStatus 2547
CancelGremlinQuery 2549
_____ 2550
ExecuteOpenCypherQuery 2550
ExecuteOpenCypherExplainQuery 2552
ListOpenCypherQueries 2554
GetOpenCypherQueryStatus 2555
CancelOpenCypherQuery 2557
_____ 2558
QueryEvalStats 2558
GremlinQueryStatus 2559
GremlinQueryStatusAttributes 2559
Chargeur en bloc 2559
StartLoaderJob 2560
GetLoaderJobStatus 2567
ListLoaderJobs 2570
CancelLoaderJob 2571
_____ 2572
LoaderIdResult 2572
Streams 2573
GetPropertygraphStream 2573
_____ 2576
PropertygraphRecord 2576
PropertygraphData 2577
Statistiques 2578
GetPropertygraphStatistics 2579
ManagePropertygraphStatistics 2580
DeletePropertygraphStatistics 2581
GetPropertygraphSummary 2582
_____ 2584

Statistiques 2584
StatisticsSummary 2585
DeleteStatisticsValueMap 2585
RefreshStatisticsIdMap 2585
NodeStructure 2586
EdgeStructure 2586
SubjectStructure 2586
PropertygraphSummaryValueMap 2587
PropertygraphSummary 2587
Traitement des données ML 2589
Démarrer MLData ProcessingJob 2589
Liste MLData ProcessingJobs 2593
Obtenez MLData ProcessingJob 2594
Annuler MLData ProcessingJob 2595
_____ 2596
MlResourceDefinition 2596
MlConfigDefinition 2597
Entraînement de modèle ML 2597
Démarrer MLModel TrainingJob 2598
Liste MLModel TrainingJobs 2601
Obtenez MLModel TrainingJob 2603
Annuler MLModel TrainingJob 2604
_____ 2606
CustomModelTrainingParameters 2606
Transformation de modèle ML 2606
Démarrer MLModel TransformJob 2607
Liste MLModel TransformJobs 2610
Obtenez MLModel TransformJob 2611
Annuler MLModel TransformJob 2612
_____ 2614
CustomModelTransformParameters 2614
Point de terminaison d'inférence ML 2614
Créez MLEndpoint 2615
Liste MLEndpoints 2617
Obtenez MLEndpoint 2618
Supprimer MLEndpoint 2620

Exceptions 2621
AccessDeniedException 2622
BadRequestException 2623
BulkLoadIdNotFoundException 2623
CancelledByUserException 2624
ClientTimeoutException 2624
ConcurrentModificationException 2624
ConstraintViolationException 2625
ExpiredStreamException 2625
FailureByQueryException 2626
IllegalArgumentException 2626
InternalFailureException 2626
InvalidArgumentException 2627
InvalidNumericDataException 2627
InvalidParameterException 2628
LoadUrlAccessDeniedException 2628
MalformedQueryException 2629
MemoryLimitExceededException 2629
MethodNotAllowedException 2629
MissingParameterException 2630
MLResourceNotFoundException 2630
ParsingException 2631
PreconditionsFailedException 2631
QueryLimitExceededException 2632
QueryLimitException 2632
QueryTooLargeException 2632
ReadOnlyViolationException 2633
S3Exception 2633
ServerShutdownException 2634
StatisticsNotAvailableException 2634
StreamRecordsNotFoundException 2634
ThrottlingException 2635
TimeLimitExceededException 2635
TooManyRequestsException 2636
UnsupportedOperationException 2636
UnloadUrlAccessDeniedException 2637

..... mmdcxxxviii

Qu'est-ce qu'Amazon Neptune ?

Amazon Neptune est un service de base de données orientée graphe entièrement géré et fiable, qui facilite la création et l'exécution d'applications fonctionnant avec des jeux de données hautement connectés. Le cœur de Neptune est un moteur de base de données orientée graphe spécialisé et hautes performances. Ce moteur est optimisé pour le stockage de milliards de relations et l'interrogation du graphe avec une latence de l'ordre de quelques millisecondes. Neptune prend en charge les langages de requête de graphes de propriétés populaires Apache TinkerPop Gremlin et OpenCypher de Neo4j, ainsi que le langage de requête RDF du W3C, SPARQL. Cela vous permet de créer des requêtes qui naviguent efficacement dans les jeux de données hautement connectés. Neptune est destiné aux cas d'utilisation axés sur les graphes, comme les moteurs de recommandation, la détection des fraudes, les graphes de connaissances, la découverte de médicaments et la sécurité du réseau.

La base de données Neptune est hautement disponible, avec des répliques en lecture, une point-in-time restauration, une sauvegarde continue sur Amazon S3 et une réplication entre les zones de disponibilité. Neptune offre des fonctionnalités de sécurité des données avec prise en charge du chiffrement au repos et en transit. Neptune est entièrement géré. Vous n'avez donc plus besoin de vous soucier des tâches de gestion de base de données, comme le provisionnement de matériel, l'application de correctifs logiciels, l'installation, la configuration ou les sauvegardes.

Neptune Analytics est un moteur de base de données analytique qui complète la base de données Neptune et qui peut analyser rapidement de grandes quantités de données graphiques en mémoire pour obtenir des informations et identifier des tendances. Neptune Analytics est une solution permettant d'analyser rapidement les bases de données graphiques existantes ou les ensembles de données graphiques stockés dans un lac de données. Elle utilise des algorithmes d'analyse de graphes populaires et des requêtes analytiques à faible latence.

Pour plus d'informations sur Amazon Neptune, nous vous conseillons de commencer par les sections suivantes :

- Premiers pas avec Amazon Neptune
- Présentation des fonctionnalités Amazon Neptune

Si vous faites vos premiers pas dans le domaine des graphes ou si vous n'êtes pas encore prêt à investir dans un environnement de production Neptune complet, consultez la rubrique Commencer à

utiliser Neptune afin de découvrir comment utiliser les blocs-notes Neptune Jupyter pour apprendre et développer sans frais supplémentaires.

En outre, avant de commencer à concevoir une base de données, nous vous recommandons de consulter le GitHub référentiel Architectures de AWS référence pour l'utilisation des bases de données graphiques, dans lequel vous pourrez vous renseigner sur vos choix concernant les modèles de données graphiques et les langages de requête, et parcourir des exemples d'architectures de déploiement de référence.

Principaux composants du service

- Instance de base de données principale : prend en charge les opérations de lecture et d'écriture, et effectue toutes les modifications de données du volume de cluster. Chaque cluster de bases de données Neptune possède une instance de base de données principale qui est responsable de l'écriture (c'est-à-dire, du chargement ou de la modification) des contenus de base de données orientée graphe.
- Réplica Neptune : se connecte au même volume de stockage que l'instance de base de données principale et prend uniquement en charge les opérations de lecture. Chaque cluster de base de données Neptune peut contenir jusqu'à 15 réplicas Neptune en plus de l'instance de base de données principale. Cela procure une haute disponibilité par la localisation des réplicas Neptune dans des zones de disponibilité distinctes et une charge de distribution à partir des clients de lecture.
- Volume de cluster : les données Neptune sont stockées dans le volume de cluster, qui a été pensé pour offrir fiabilité et haute disponibilité. Un volume de cluster est constitué de copies des données réparties dans plusieurs zones de disponibilité d'une même AWS région. Comme vos données sont automatiquement répliquées dans toutes les zones de disponibilité, elles sont hautement durables et le risque de perte des données est très faible.

Supporte Open Graph APIs

Amazon Neptune prend en charge les graphes ouverts APIs pour les graphes de propriétés (Gremlin et OpenCypher) et les graphes RDF (SPARQL). Il fournit des performances élevées pour ces deux modèles de graphes et leurs langages de requête. Vous pouvez choisir le modèle Property Graph (PG) et accéder au même graphe avec le langage de requête OpenCypher et and/or le langage de requête Gremlin. Si vous utilisez le modèle RDF (Resource Description Framework) standard du W3C, vous pouvez accéder à votre graphe à l'aide du langage de requête SPARQL standard.

Hautement sécurisé

Neptune propose plusieurs niveaux de sécurité pour votre base de données. Les fonctionnalités de sécurité incluent l'isolement réseau avec Amazon VPC, ainsi que le chiffrement au repos avec des clés que vous créez et contrôlez via AWS Key Management Service (AWS KMS). Dans une instance Neptune chiffrée, les données du stockage sous-jacent sont chiffrées, de même que les sauvegardes automatiques, les instantanés et les réplicas dans le même cluster.

Entièrement géré

Grâce à Amazon Neptune, vous n'avez plus besoin de vous soucier des tâches de gestion de base de données, comme le provisionnement de matériel, l'application de correctifs logiciels, l'installation, la configuration et les sauvegardes.

Vous pouvez utiliser Neptune pour créer des applications de graphe sophistiquées interactives capables d'interroger des milliards de relations en quelques millisecondes. Les requêtes SQL pour les données hautement connectées sont complexes et difficiles à ajuster pour obtenir des performances optimales. Neptune vous permet d'utiliser les langages de requête de graphe populaires Gremlin, openCypher et SPARQL pour exécuter des requêtes puissantes qui sont faciles à écrire et efficaces avec des données connectées. Cette fonctionnalité réduit considérablement la complexité du code, ce qui vous permet de créer rapidement des applications qui traitent les relations.

Neptune est conçu pour offrir une disponibilité supérieure à 99,99 %. Ce service améliore les performances et la disponibilité des bases de données en intégrant au moteur de base de données une couche de stockage SSD virtualisé conçue pour les charges de travail de base de données. Stockage tolérant aux pannes et auto-réparable. Les défaillances de disque sont réparées en arrière-plan sans perte de disponibilité de la base de données. Neptune détecte automatiquement les incidents de base de données et redémarre sans effectuer de récupération sur incident et sans regénérer le cache de la base de données. Si l'instance échoue dans son ensemble, Neptune bascule automatiquement vers l'un des 15 réplicas en lecture.

Modifications et mises à jour apportées à Amazon Neptune

Le tableau suivant décrit les modifications importantes apportées à Amazon Neptune.

Modification	Description	Date
Neptune dans la région Asie-Pacifique (Hyderabad)	Amazon Neptune est désormais disponible dans la région Asie-Pacifique (Hyderabad) (). `ap-south-2`	10 mars 2026
Version du moteur 1.4.7.0	Depuis le 03/03/2020, la version 1.4.7.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.4.7.0 de Neptune Engine.	3 mars 2026
Version du moteur 1.4.6.3	Depuis le 18 décembre 2025, la version 1.4.6.3 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.4.6.3 du moteur Neptune.	18 décembre 2025
Version du moteur 1.4.6.2	Depuis le 18 novembre 2025, la version 1.4.6.2 du moteur	18 novembre 2025

	est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.4.6.2 du moteur Neptune.	
Version du moteur 1.4.6.1	Depuis le 18/09/2025, la version 1.4.6.1 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.4.6.1 du moteur Neptune.	18 septembre 2025
Version du moteur 1.4.6.0	Depuis le 02/09/2020, la version 1.4.6.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez Neptune Engine version 1.4.6.0.	2 septembre 2025

Points de terminaison publics	Amazon Neptune prend désormais en charge les points de terminaison publics. Pour plus d'informations, consultez la section Points de terminaison publics.	27 août 2025
Version du moteur 1.4.5.1	Au 30 juin 2020, la version 1.4.5.1 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.4.5.1 du moteur Neptune.	30 juin 2025
Neptune a été lancé dans les régions Asie-Pacifique (Melbourne) et Canada-Ouest (Calgary)	Amazon Neptune est désormais disponible dans les régions Asie-Pacifique (Melbourne) (`ap-southeast-4`) et Canada Ouest (Calgary) (`ca-west-1`).	28 mai 2025
Version du moteur 1.4.5.0	Depuis le 2025-04-09, la version 1.4.5.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.4.5.0 de Neptune Engine.	9 avril 2025

Neptune a été lancé dans la région Asie-Pacifique (Malaisie)	Amazon Neptune est désormais disponible dans la région Asie-Pacifique (Malaisie) (`ap-southeast-5`). Les blocs-notes gérés ne sont pas pris en charge pour la région Asie-Pacifique (Malaisie) pour le moment, mais ils peuvent être configurés manuellement. Voir Configuration manuelle des blocs-notes Neptune pour plus d'informations.	28 février 2025
Version du moteur 1.4.4.0	Au 24/02/2025, la version 1.4.4.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.4.4.0 du moteur Neptune.	24 février 2025
Version du moteur 1.4.3.0	Depuis le 21/01/2021, la version 1.4.3.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.4.3.0 du moteur Neptune.	21 janvier 2025

Version du moteur 1.4.2.0	Depuis le 19 décembre 2024-12, la version 1.4.2.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.4.2.0 de Neptune Engine.	19 décembre 2024
Version du moteur 1.4.1.0	Depuis le 21/11/2021, la version 1.4.1.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.4.1.0 de Neptune Engine.	21 novembre 2024
Version du moteur 1.4.0.0	Depuis le 06/11/2021, la version 1.4.0.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.4.0.0 de Neptune Engine.	6 novembre 2024

Version du moteur 1.3.4.0	Depuis le 2024-10-01, la version 1.3.4.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.3.4.0 du moteur Neptune.	1er octobre 2024
Version du moteur 1.3.3.0	Depuis le 5 août 2021, la version 1.3.3.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.3.3.0 du moteur Neptune.	5 août 2024
Version du moteur 1.2.1.2	Depuis le 5 août 2021, la version 1.2.1.2 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.2.1.2 du moteur Neptune.	5 août 2024

Version du moteur 1.3.2.1	Depuis le 2024-06-20, la version 1.3.2.1 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.3.2.1 du moteur Neptune.	20 juin 2024
Version du moteur 1.3.2.0	Depuis le 10/06/2021, la version 1.3.2.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.3.2.0 de Neptune Engine.	10 juin 2024
Version du moteur 1.2.1.1	Depuis le 11/03/2021, la version 1.2.1.1 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.2.1.1 du moteur Neptune.	11 mars 2024

Version du moteur 1.3.1.0	Depuis le 06/03/2021, la version 1.3.1.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.3.1.0 de Neptune Engine.	6 mars 2024
Mise à jour des autorisations de politique AWS gérées	Les politiques `NeptuneFullAccess` gérées `NeptuneReadOnlyAcess` et incluent désormais `Sid` (ID de déclaration) comme identifiant dans la déclaration de politique.	22 janvier 2024
Neptune propose désormais un stockage optimisé pour les E/S	Avec I/O-Optimized storage, you pay for the storage and instances you are using. The storage costs are higher than for standard storage, but you pay nothing for any I/O ça, tu utilises.	13 décembre 2023

Modifications de la politique gérée par IAM pour Neptune	La politique gérée par NeptuneConsoleFullAccessIAM a été mise à jour pour accorder les autorisations nécessaires pour interagir avec les graphes de Neptune Analytics, une NeptuneGraphReadOnlyAccessnouvelle politique a été ajoutée pour fournir un accès en lecture seule aux ressources graphiques de Neptune Analytics, et `AWSServiceRoleForNeptuneGraphPolicy` une nouvelle politique a été ajoutée pour permettre aux graphiques CloudWatch de Neptune Analytics de publier des statistiques et des journaux opérationnels et d'utilisation.	29 novembre 2023
Version du moteur 1.3.0.0	Depuis le 15-11-2023, la version 1.3.0.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez Version 1.3.0.0 du moteur Neptune.	15 novembre 2023

Neptune a été lancé dans la région Israël (Tel Aviv)	Amazon Neptune est désormais disponible dans la région Israël (Tel Aviv) (`il-central-1`).	13 novembre 2023
Article de blog sur l'implémentation time-to-live dans les graphes de propriétés Neptune	Consultez Implement Time to Live in Amazon Neptune, Part 1: Property Graph (Implémenter la durée de vie dans Amazon Neptune, partie 1 : graphe de propriété) de Melissa Kwok, Mike Havey et Kevin Phillips.	27 octobre 2023
Version de moteur 1.2.1.0.R7	Depuis le 06-10-2023, la version 1.2.1.0.R7 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.1.0.R7 du moteur Neptune.	6 octobre 2023

Version de moteur 1.2.0.0.R4	Depuis le 29 septembre 2023, la version 1.2.0.0.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.0.0.R4 du moteur Neptune.	29 septembre 2023
Version de moteur 1.2.0.1.R3	Depuis le 27 septembre 2023, la version 1.2.0.1.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.0.1.R3 du moteur Neptune.	27 septembre 2023

Version de moteur 1.2.1.0.R6	Depuis le 12 septembre 2023, la version 1.2.1.0.R6 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.1.0.R6 du moteur Neptune.	12 septembre 2023
Version de moteur 1.2.0.2.R6	Depuis le 12 septembre 2023, la version 1.2.0.2.R6 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.0.2.R6 du moteur Neptune.	12 septembre 2023
Article de blog sur l'utilisation d'une stratégie blue/green de déploiement pour les mises à niveau du moteur Neptune	Consultez Améliorer la disponibilité d'Amazon Neptune lors de la mise à niveau du moteur grâce au blue/green déploiement réalisé par Ankit Gupta et Abhishek Mishra.	11 septembre 2023

Version de moteur 1.2.1.0.R5	Depuis le 2 septembre 2023, la version 1.2.1.0.R5 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la version 1.2.1.0.R5 du moteur Neptune.	2 septembre 2023
Version de moteur 1.2.0.2.R5	Depuis le 16 août 2023, la version 1.2.0.2.R5 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.0.2.R5 du moteur Neptune.	16 août 2023
Version de moteur 1.2.1.0.R4	Depuis le 10 août 2023, la version 1.2.1.0.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.1.0.R4 du moteur Neptune.	10 août 2023

Billet de blog sur la version 1.2.1.0 du moteur Neptune	Consultez Exploring the feature packed 1.2.1.0 release for Amazon Neptune (Explorer la version 1.2.1.0 riche en fonctionnalités d'Amazon Neptune) de Joy Wang, Kevin Phillips, Andrea Nassisi et Navtanay Sinha.	4 août 2023
Billet de blog sur la création d'une solution de base de données multimodale avec Neptune	Consultez Design a use case-driven, highly scalable multimodel database solution using Amazon Neptune (Concevoir une solution de base de données multimodèle hautement évolutive et basée sur des cas d'utilisation à l'aide d'Amazon Neptune) de Mike Havey.	18 juillet 2023
Billet de blog sur les cas d'utilisation et les bonnes pratiques de Neptune sans serveur	Consultez Use cases and best practices to optimize cost and performance with Amazon Neptune Serverless (Cas d'utilisation et bonnes pratiques permettant d'optimiser les coûts et les performances avec Amazon Neptune sans serveur) de Kevin Phillips et Ankit Gupta.	28 juin 2023

Version de moteur 1.2.1.0.R3	Depuis le 13 juin 2023, la version 1.2.1.0.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.1.0.R3 du moteur Neptune.	13 juin 2023
Billet de blog sur la génération de suggestions de loisirs en temps réel	Consultez Generate suggestions for leisure activities in real time with Amazon Neptune (Générer des suggestions d'activités de loisirs en temps réel avec Amazon Neptune) de Michael Meidlinger et Nils Müller.	6 juin 2023
Article de blog sur la modélisation moléculaire avec Neptune et RDKit	Voir les données du modèle moléculaire SMILES avec Amazon Neptune et RDKit par Graham Kutchek.	1er juin 2023
Billet de blog sur l'utilisation de la mise en cache pour accélérer les performances de Neptune (partie 3)	Consultez Accélérez les performances des requêtes graphiques grâce à la mise en cache dans Amazon Neptune, partie 3 : Architectures de mise en cache à l'échelle du cluster Neptune avec ElastiCache Amazon par Taylor Riggan, Abhishek Mishra, Melissa Kwok et Kelvin Lawrence.	26 mai 2023

Billet de blog sur l'utilisation de la mise en cache pour accélérer les performances de Neptune (partie 2)	Consultez Accelerate graph query performance with caching in Amazon Neptune, Part 2: Additional Neptune caching features (Accélérer les performances des requêtes orientées graphe grâce à la mise en cache dans Amazon Neptune, partie 2 : fonctionn alités supplémentaires de mise en cache Neptune) de Taylor Riggan, Abhishek Mishra, Melissa Kwok et Kelvin Lawrence.	26 mai 2023
Billet de blog sur l'utilisation de la mise en cache pour accélérer les performances de Neptune (partie 1)	Consultez Accelerate graph query performance with caching in Amazon Neptune, Part 1: Queries and buffer pool caching (Accélérer les performances des requêtes orientées graphe grâce à la mise en cache dans Amazon Neptune, partie 1 : requêtes et mise en cache du pool de mémoire tampon) de Taylor Riggan, Abhishek Mishra, Melissa Kwok et Kelvin Lawrence.	26 mai 2023
Billet de blog sur l'analyse de la chaîne d'approvisionnement à l'aide de Neptune	Consultez le billet de blog Supply chain data analysis and visualization using Amazon Neptune and the Neptune workbench de Dhiraj Thakur et Rajdip Chaudhur.	10 mai 2023

Version de moteur 1.2.0.2.R4	Depuis le 8 mai 2023, la version 1.2.0.2.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.0.2.R4 du moteur Neptune.	8 mai 2023
Lancement de Neptune dans la région Moyen-Orient (EAU)	Amazon Neptune est désormais disponible dans la région du Moyen-Orient (UAE) (`me-central-1`).	2 mai 2023
Version de moteur 1.2.1.0.R2	Depuis le 2 mai 2023, la version 1.2.1.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.1.0.R2 du moteur Neptune.	2 mai 2023

Billet de blog sur la création d'un graphe de connaissances sur Neptune avec une analyse vidéo basée sur l'IA à l'aide de Media2Cloud	Consultez Build a knowledge graph on Amazon Neptune with AI-powered video analysis using Media2Cloud (Créer un graphe de connaissances sur Amazon Neptune grâce à une analyse vidéo basée sur l'IA à l'aide de Media2Cloud) de Mike Havey.	2 mai 2023
Article de blog expliquant comment créer une DevOcean plateforme de correction des vulnérabilités à l'aide de Neptune	Découvrez How DevOcean a créé une plateforme de gestion de la correction des vulnérabilités pour les applications cloud natives à l'aide d'Amazon Neptune par Gil Makmel et Charles Ivie.	25 avril 2023
Billet de blog expliquant comment Getir a créé un système de détection des fraudes à l'aide de Neptune	Consultez le billet de blog How Getir build a comprehensive fraud detection system using Amazon Neptune and Amazon DynamoDB de Berkay Berkman, Mahmut Turan, Mutlu Polatcan, Umut Cemal Kıraç, Yağız Yanıkoğlu et Esra Kayabali.	6 avril 2023
Billet de blog expliquant comment Wiz réinvente la sécurité du cloud à l'aide de Neptune	Consultez The World is a graph: How Wiz reimagines cloud security using a graph in Amazon Neptune (Le monde est un graphe : comment Wiz réinvente la sécurité du cloud à l'aide d'un graphe dans Amazon Neptune d'Ami Luttwak et Brad Bebee.	31 mars 2023

Version de moteur 1.2.0.2.R3	Depuis le 27 mars 2023, la version 1.2.0.2.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.0.2.R3 du moteur Neptune.	27 mars 2023
Billet de blog expliquant comment CSC Generation favorise la découverte de produits à l'aide de Neptune	Consultez le billet de blog How CSC Generation powers product discovery with knowledge graphs using Amazon Neptune de Bobber Cheng, Ronit Rudra et Melissa Kwok.	21 mars 2023
Version du moteur 1.2.1.0	Depuis le 8 mars 2023, la version 1.2.1.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.1.0 du moteur Neptune.	8 mars 2023
Article de blog sur l'exploration des nouvelles fonctionnalités de la version TinkerPop 3.6.x dans Neptune	Consultez la section Exploration des nouvelles fonctionnalités d'Apache TinkerPop 3.6.x dans Amazon Neptune par Stephen Mallette.	8 mars 2023

Billet de blog sur l'utilisation du raisonnement sémantique pour déduire de nouveaux faits à partir de votre graphe RDF	Consultez Utiliser un raisonnement sémantique pour déduire de nouveaux faits à partir de votre graphe RDF en l'intégrant à Amazon RDFox Neptune de Charles Ivie et Diana Marks.	20 février 2023
Billet de blog sur l'analyse des données FHIR de soins de santé avec Neptune	Consultez le billet de blog Analyze healthcare FHIR data with Amazon Neptune d'Alena Schmickl.	13 février 2023
Billet de blog sur la création d'une solution de détection des fraudes en temps réel à l'aide de Neptune ML	Consultez le billet de blog Build a real-time fraud detection solution using Amazon Neptune ML de Hua Shu et Soji Adeshina.	8 février 2023
Version de moteur 1.1.1.0.R7	Depuis le 23 janvier 2023, la version 1.1.1.0.R7 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.1.1.0.R7 du moteur Neptune.	23 janvier 2023
Sortie de graph-explorer	Graph-explorer est un outil d'application web frontal open source permettant de visualiser les données de graphe. Voir https://github.com/aws/graph-explorer.	3 janvier 2023

Version de maintenance 1.1.0.0.R3	Depuis le 23 décembre 2022, la version de maintenance 1.1.0.0.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.1.0.0.R3 du moteur Neptune.	23 décembre 2022
Neptune Workbench fonctionne désormais sur Amazon Linux 2 et 3. JupyterLab	Les blocs-notes Neptune Graph fonctionnent désormais dans un environnement Amazon Linux 2 avec 3. JupyterLab Consultez la section Migration de vos blocs-notes Neptune de Jupyter JupyterLab vers 3 pour savoir comment migrer vers ce nouvel environnement.	21 décembre 2022
Article de blog sur les recommandations énergétiques et la recherche à l'aide d'un graphe de IMDb connaissances (partie 3)	Consultez les recommandations de Power et effectuez une recherche à l'aide d'un graphe de IMDb connaissances — 3e partie par Divya Bhargavi, Soji Adeshina, Gaurav Rele, Karan Sindwani, Vidya Sagar Ravipati et Matthew Rhodes.	20 décembre 2022

Article de blog sur les recommandations énergétiques et la recherche à l'aide d'un graphe de IMDb connaissances (partie 2)	Consultez les recommandations de Power et effectuez une recherche à l'aide d'un graphe de IMDb connaissances — Partie 2 par Matthew Rhodes, Soji Adeshina, Divya Bhargavi, Gaurav Rele, Karan Sindwani et Vidya Sagar Ravipati.	20 décembre 2022
Article de blog sur les recommandations énergétiques et la recherche à l'aide d'un graphe de IMDb connaissances (partie 1)	Consultez les recommandations de Power et effectuez une recherche à l'aide d'un graphe de IMDb connaissances — Partie 1 par Gaurav Rele, Soji Adeshina, Divya Bhargavi, Karan Sindwani, Vidya Sagar Ravipati et Matthew Rhodes.	20 décembre 2022
Neptune Serverless est désormais disponible dans de nouvelles régions AWS	Le 16 décembre 2022, Neptune sans serveur a été lancé dans les nouvelles régions AWS suivantes : Canada (Centre), Europe (Stockholm), Europe (Francfort), Asie-Pacifique (Singapour) et Asie-Pacifique (Sydney). Consultez les contraintes d'Amazon Neptune sans serveur pour vous familiariser avec toutes les régions dans lesquelles Neptune sans serveur est disponible.	16 décembre 2022

Version de moteur 1.2.0.2.R2	Depuis le 15 décembre 2022, la version 1.2.0.2.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.0.2.R2 du moteur Neptune.	15 décembre 2022
Version de moteur 1.2.0.0.R3	Depuis le 15 décembre 2022, la version 1.2.0.0.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.0.0.R3 du moteur Neptune.	15 décembre 2022

Version de moteur 1.2.0.1.R2	Depuis le 13 décembre 2022, la version 1.2.0.1.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.0.1.R2 du moteur Neptune.	13 décembre 2022
Article de blog sur la conception d'une architecture pédagogique d'analyse des mégadonnées avec AWS	Consultez le billet de blog Designing an educational big data analysis architecture with AWS de Lavanya Sood.	13 décembre 2022
Billet de blog expliquant comment les bases de données orientées graphe peuvent améliorer l'apprentissage	Consultez le billet de blog How graph databases can enhance learning de Lavanya Sood.	8 décembre 2022
Article de blog sur le chargement de données RDF dans AWS Neptune à l'aide de Glue	Consultez Charger des données RDF dans Amazon Neptune with AWS Glue de Mike Havey et Fabrizio Napolitano.	23 novembre 2022

Version du moteur 1.2.0.2	Depuis le 20 novembre 2022, la version 1.2.0.2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.0.2 du moteur Neptune.	20 novembre 2022
Version du moteur 1.2.0.1	Depuis le 26 octobre 2022, la version 1.2.0.1 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.0.1 du moteur Neptune.	26 octobre 2022
Billet de blog sur la détection des fraudes avec Neptune	Consultez le billet de blog Empowering fraud detection at Delivery Hero with Amazon Neptune de Wilson Tang, Amr Elnaggar, Matias Pons, Mohammad Azzam, Saurabh Deshpande et Luis Rodrigues Soares.	26 octobre 2022

Billet de blog sur Neptune sans serveur	Consultez le billet de blog Introducing Amazon Neptune Serverless – A Fully Managed Graph Database that Adjusts Capacity for Your Workloads de Danilo Poccia.	26 octobre 2022
Billet de blog sur les importations RDF pilotées par des événements dans Neptune à l'aide de Lambda et SPARQL UPDATE LOAD	Découvrez comment NXP effectue des importations RDF basées sur des événements vers Amazon Neptune à l'aide de AWS Lambda et de SPARQL UPDATE LOAD de John Walker, Onno Buijs, Charles Ivie et Javy de Koning.	20 octobre 2022
Version de moteur 1.2.0.0.R2	Depuis le 14 octobre 2022, la version 1.2.0.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.0.0.R2 du moteur Neptune.	14 octobre 2022
Billet de blog sur l'encodage des propriétés de texte multilingues dans Neptune	Consultez le billet de blog Encode multi-lingual text properties in Amazon Neptune to train predictive models de Jiani Zhang.	14 octobre 2022

Billet de blog sur les tests automatisés de l'accès aux données Neptune	Voir Tests automatisés de l'accès aux données d'Amazon Neptune avec Apache TinkerPop Gremlin par Greg Biegel.	28 septembre 2022
Version de moteur 1.1.1.0.R6	Depuis le 23 septembre 2022, la version 1.1.1.0.R6 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.1.1.0.R6 du moteur Neptune.	23 septembre 2022
Billet de blog sur la façon dont Informatica® utilise Neptune	Consultez le billet de blog How Informatica® Cloud Data Governance and Catalog uses Amazon Neptune for knowledge graphs de Tiju Titus John, Deepak Ram et Farooq Ashraf.	20 septembre 2022
Billet de blog sur l'utilisation de Neptune et de Tom Sawyer Perspectives pour détecter des fraudes financières	Consultez le billet de blog Uncover financial fraud with Amazon Neptune and Tom Sawyer Perspectives de Janet M. Six, chef de produit senior chez Tom Sawyer Software.	30 août 2022

Article de blog sur la création d'une solution de détection des fraudes en temps réel basée sur le GNN à l'aide de SageMaker Neptune et de DGL	Découvrez Créer une solution de détection des fraudes en temps réel basée sur GNN à l'aide d'Amazon SageMaker , Amazon Neptune et de la bibliothèque Deep Graph de Jian Zhang, Haozhu Wang et Mengxin Zhu.	11 août 2022
Billet de blog sur l'utilisation de balises de ressources pour arrêter et démarrer les ressources de l'environnement Neptune	Consultez le billet de blog Automate the stopping and starting of Amazon Neptune environment resources using resource tags de Kevin Phillips.	1er août 2022
Article de blog sur un artiste qui contribue à Apache TinkerPop	Découvrez Beyond Code : L'artiste qui contribue à Apache de TinkerPop Stephen Mallette et Ketrina Thompson.	1er août 2022
Billet de blog sur le contrôle d'accès précis pour les actions du plan de données Neptune	Consultez le billet de blog Fine Grained Access Control for Amazon Neptune data plane actions d'Abhishek Mishra et Ankit Gupta.	29 juillet 2022
Billet de blog sur les bases de données globales Neptune	Consultez le billet de blog Introducing Amazon Neptune Global Database de Navtanay Sinha.	27 juillet 2022

Version du moteur 1.2.0.0	Depuis le 21 juillet 2022, la version 1.2.0.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.2.0.0 du moteur Neptune.	21 juillet 2022
Version de moteur 1.1.1.0.R5	Depuis le 21 juillet 2022, la version 1.1.1.0.R5 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.1.1.0.R5 du moteur Neptune.	21 juillet 2022
Version de moteur 1.1.1.0.R4	Depuis le 23 juin 2022, la version 1.1.1.0.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.1.1.0.R4 du moteur Neptune.	23 juin 2022

Simplification des flux de travail d'analyses et de machine learning orientés graphe grâce à l'intégration de Python	Vous pouvez désormais exécuter des tâches de machine learning et d'analyti que de graphe sur des données de graphe stockées dans Amazon Neptune à l'aide d'une intégration Python open source qui simplifie les flux de travail liés à la science des données et au machine learning. Consultez la documentation AWS Data Wrangler pour Neptune.	7 juin 2022
Version de moteur 1.1.1.0.R3	Depuis le 7 juin 2022, la version 1.1.1.0.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.1.1.0.R3 du moteur Neptune.	7 juin 2022
Billet de blog sur la détection des fausses nouvelles	Consultez le billet de blog Detect social media fake news using graph machine learning with Amazon Neptune ML de Hasan Shojaei et Sarita Joshi.	19 mai 2022

Billet de blog sur l'utilisation de SQL Server Integration Services (SSIS) avec Neptune	Consultez le billet de blog Discover new insights from your data using SQL Server Integration Services (SSIS) and Amazon Neptune de Mesgana Gormley et Melissa Kwok.	18 mai 2022
Version de maintenance 1.1.1.0.R2	Depuis le 16 mai 2022, la version de maintenance 1.1.1.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.1.1.0.R2 du moteur Neptune.	16 mai 2022
Version de maintenance 1.1.0.0.R2	Depuis le 16 mai 2022, la version de maintenance 1.1.0.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.1.0.0.R2 du moteur Neptune.	16 mai 2022

Version de maintenance 1.0.5.1.R4	Depuis le 16 mai 2022, la version de maintenance 1.0.5.1.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.5.1.R4 du moteur Neptune.	16 mai 2022
Version de maintenance 1.0.5.0.R5	Depuis le 16 mai 2022, la version de maintenance 1.0.5.0.R5 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.5.0.R5 du moteur Neptune.	16 mai 2022
Billet de blog sur la disponibilité générale d'openCypher dans Neptune	Consultez le billet de blog Announcing the General Availability of openCypher support for Amazon Neptune de Navtanay Sinha et Dave Bechberger.	22 avril 2022

Version du moteur 1.1.1.0	Depuis le 19 avril 2022, la version 1.1.1.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.1.1.0 du moteur Neptune.	19 avril 2022
Billet de blog sur le lignage des données	Voir Créer un lignage de données pour les lacs de données à l'aide de AWS Glue, Amazon Neptune et Spline, par Khoa Nguyen, Krithivasan Balasubramaniyan et Rahul Shaurya.	1er avril 2022
Les mises à niveau de la version 1.1.0.0 du moteur sont réactivées	Le 21 février 2022, les mises à niveau de la version 1.1.0.0 du moteur ont été temporairement désactivées. Elles sont maintenant réactivés.	22 mars 2022
Billet de blog sur l'amélioration de la fiabilité des services publics	Consultez le billet de blog Using cloud-based, data-informed, power system models to engineer utility reliability d'Abhineet Parchure.	22 mars 2022

Lancement de Neptune dans la région Afrique (Le Cap)	Amazon Neptune est désormais disponible dans la région Afrique (Le Cap) (`af-south-1`). Toutefois, la prise en charge des bloc-notes du workbench Neptune est temporairement désactivée sur la console Neptune dans cette région.	24 février 2022
Billet de blog sur les graphes axés sur les modèles avec OWL	Consultez le billet de blog Model-driven graphs using OWL in Amazon Neptune de Mike Havey.	23 février 2022
Billet de blog sur l'exploration des graphes de connaissances sémantiques avec Rhizomer	Consultez le billet de blog Explore the semantic knowledge graphs without SPARQL using Amazon Neptune with Rhizomer de Roberto García.	22 février 2022
Billet de blog sur la création de graphe pour le réseau électrique	Voir Graphing the utility grid on AWS, par Bobby Wilson et Joseph Beer.	18 février 2022
Nouvelles options d'encodage des fonctionnalités de texte de Neptune ML	Neptune prend désormais en charge le codage FastText de texte Sentence BERT pour l'entraînement. Découvrez les FastTextfonctionnalités de Neptune ML et les fonctionnalités de Sentence BERT dans Neptune ML.	15 février 2022

Article de blog sur l'utilisation de requêtes géospatiales OpenSearch avec Neptune	Voir Combiner Amazon Neptune et Amazon OpenSearch Service pour les requêtes géospatiales, par Ross Gabay et Abhilash Vinod.	1er février 2022
Billet de blog sur la détection des crimes financiers à l'aide d'Amazon EKS et Neptune	Consultez le billet de blog Financial Crime Discovery using Amazon EKS and Graph Databases de Severin Gassauer-Fleissner et Zahi Ben Shabat.	1er février 2022
Un volume de cluster Neptune peut désormais atteindre une taille de 128 tébioctets (TiO)	Dans toutes les régions prises en charge GovCloud, à l'exception de la Chine et de la Chine, la limite de taille d'un volume de cluster Neptune est désormais passée de 64 TiB à 128 TiB. Cette limite s'applique à toutes les versions du moteur à partir de la version 1.0.2.2. Consultez la page Stockage Amazon Neptune.	1er février 2022
Neptune intègre désormais la recherche en texte intégral à toutes les versions de. OpenSearch	Consultez la section Recherche en texte intégral dans Amazon Neptune à l'aide d'Amazon OpenSearch Service.	28 janvier 2022

Billet de blog sur l'utilisation de conteneurs Docker pour déployer des blocs-notes de graphe	Voir Utiliser des conteneurs Docker pour déployer des blocs-notes Graph sur AWS, par Ganesh Sawhney et Qiang Zhang.	22 janvier 2022
Version de moteur 1.0.5.1.R3	Depuis le 13 janvier 2022, la version 1.0.5.1.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.5.1.R3 du moteur Neptune.	13 janvier 2022
Billet de blog sur un système de recommandation basé sur des graphes utilisant Neptune ML	Consultez le billet de blog Graph-based recommendation system with Neptune ML: An illustration on social network link prediction challenges de Yanwei Cui et Will Badr.	12 janvier 2022
Billet de blog sur l'autoscaling Neptune	Consultez le billet de blog Auto scale your Amazon Neptune database to meet workload demands de Navtanay Sinha et Sudhanshu Gupta.	29 novembre 2021

Billet de blog sur l'analyse et la visualisation de données de graphe interactives	Consultez Créer des analyses et des visualisations de données graphiques interactives à l'aide d'Amazon Neptune, Amazon Athena Federated Query et Amazon, par Sandeep Veldi et QuickSight Abhishek Mishra.	24 novembre 2021
Billet de blog sur la détection de l'usurpation d'identité à l'aide du deep learning basé sur des graphes	Consultez le billet de blog How Careem is detecting identity fraud using graph-based deep learning and Amazon Neptune de Kevin O'Brien, Kamran Habib et Will Badr.	23 novembre 2021
Version du moteur 1.1.0.0	Depuis le 19 novembre 2021, la version 1.1.1.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.1.0.0 du moteur Neptune.	19 novembre 2021
Billet de blog sur la centralisation de la protection des données et de la conformité	Consultez la section Centralisation de la protection et de la conformité des données dans Amazon Neptune with AWS Backup, par Brian O'Keefe.	8 novembre 2021

Billet de blog sur la lutte contre la fraude et les paiements irréguliers	Consultez le billet de blog Fighting fraud and improper payments in real-time at the scale of federal expenditures de Vladi Royzman et Spencer Smith.	2 novembre 2021
Billet de blog sur la prévention de la création de faux comptes	Consultez le billet de blog Prevent fake account sign-ups in real time with AI using Amazon Fraud Detector d'Anjan Biswas.	29 octobre 2021
Version de moteur 1.0.5.1.R2	Depuis le 26 octobre 2021, la version 1.0.5.1.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.5.1.R2 du moteur Neptune.	26 octobre 2021
Article de blog sur la prévision à HawkEye 360° des risques liés aux navires à l'aide de la bibliothèque Deep Graph	See HawkEye 360 prédit les risques liés aux navires à l'aide de la Deep Graph Library et d'Amazon Neptune de Tim Pavlick, Ian Avilez, Dan Ford et Gaurav Rele.	15 octobre 2021

Version du moteur 1.0.5.1	Depuis le 1er octobre 2021, la version 1.0.5.1 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.5.1 du moteur Neptune.	1er octobre 2021
Article de blog expliquant pourquoi les développeurs aiment TinkerPop	Découvrez pourquoi les développeurs aiment Apache TinkerPop, un framework open source pour le calcul graphique, par Brad Bebee, Kelvin Lawrence et Stephen Mallette.	27 septembre 2021
Version de moteur 1.0.5.0.R3	Depuis le 15 septembre 2021, la version 1.0.5.0.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessair es pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.5.0.R3 du moteur Neptune.	15 septembre 2021

Billet de blog sur la création d'une solution de découverte de données avec Amundsen et Neptune	Consultez le billet de blog Building a data discovery solution with Amundsen and Amazon Neptune de Peter Hanssens et Don Simpson.	8 septembre 2021
Neptune a mis à jour l'interrogateur de flux pour prendre en charge les requêtes de recherche en texte intégral sans chaînes	Cette version inclut de nombreuses améliorations apportées à la recherche en texte intégral, notamment la prise en charge de l'indexation des valeurs de propriétés qui ne sont pas des chaînes. Consultez la section OpenSearch Indexation autre que des chaînes dans Amazon Neptune.	23 août 2021
Version de moteur 1.0.5.0.R2	Depuis le 16 août 2021, la version 1.0.5.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.5.0.R2 du moteur Neptune.	16 août 2021

Version de moteur 1.0.4.2.R5	Depuis le 16 août 2021, la version 1.0.4.2.R5 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.4.2.R5 du moteur Neptune.	16 août 2021
Billet de blog sur la prise en charge du protocole Graph Store dans Neptune	Consultez le billet de blog Introducing Graph Store Protocol support for Amazon Neptune de Chris Smith.	2 août 2021
Billet de blog sur les nouvelles fonctionnalités de Neptune ML	Consultez le billet de blog Discover more insights in your graphs with new features from Amazon Neptune ML de Soji Adeshina.	30 juillet 2021
Billet de blog sur l'accélération des prédictions avec Neptune ML	Consultez le billet de blog Get predictions for evolving graph data faster with Amazon Neptune ML de Soji Adeshina.	30 juillet 2021
Billet de blog sur la simplication et l'accélération du machine learning des graphes avec Neptune ML	Consultez le billet de blog Easier and faster graph machine learning with Amazon Neptune ML de Soji Adeshina.	30 juillet 2021

Billet de blog sur la prise en charge d'openCypher pour Neptune	Consultez le billet de blog Announcing openCypher for Amazon Neptune: Building better graph applications with openCypher and Gremlin together de Brad Bebee.	29 juillet 2021
Version du moteur 1.0.5.0	Depuis le 27 juillet 2021, la version 1.0.5.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.5.0 du moteur Neptune.	27 Juillet 2021
Version de moteur 1.0.4.2.R4	Depuis le 23 juillet 2021, la version 1.0.4.2.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.4.2.R4 du moteur Neptune.	23 juillet 2021
Lancement de Neptune dans la région Chine (Beijing)	Amazon Neptune est désormais disponible en Chine (Beijing) (`cn-north-1`).	21 juillet 2021

Version de moteur 1.0.4.2.R3	Depuis le 28 juin 2021, la version 1.0.4.2.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.4.2.R3 du moteur Neptune.	28 juin 2021
Billet de blog expliquant comment Dream11 a étendu son réseau social à l'aide de Neptune	Découvrez comment Dream11, la plus grande plateforme de sports fantastiques au monde, développe son réseau social avec Amazon Neptune et Amazon. ElastiCache	25 juin 2021
Article de blog sur la transformation des données en connaissances avec Neptune et Semantic Suite PoolParty	Consultez Transformer les données en connaissances avec PoolParty Semantic Suite et Amazon Neptune, par Ioanna Lytra et Albin Ahmeti.	16 juin 2021
Article de blog sur l'utilisation de Neptune pour explorer la base de connaissances UniProt	Voir Exploration de la base de connaissances sur les UniProt protéines avec AWS Open Data et Amazon Neptune, par Eric Greene, Rafa Xu et Yuan Shi.	10 juin 2021

Billet de blog sur l'utilisation de Neptune pour l'analyse des risques basée sur les données	Voir les notes de terrain : Analyse des risques basée sur les données avec Amazon Neptune et OpenSearch Amazon Service, par Adriaan de Jonge et Rohit Satyanara yana.	10 juin 2021
Version de moteur 1.0.4.2.R2	Depuis le 1er juin 2021, la version 1.0.4.2.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.4.2.R2 du moteur Neptune.	1er juin 2021
Article de blog sur l'utilisation de Neptune pour visualiser votre infrastructure AWS	Consultez Visualisez votre AWS infrastructure avec Amazon Neptune et AWS Config, par Rohan Raizada et Amey Dhavle.	25 mai 2021
Billet de blog sur la configura tion pour l'utilisation de Data Lens avec Neptune	Consultez Configurer les AWS services pour créer un graphe de connaissances dans Amazon Neptune à l'aide de Data Lens, par Russell Waterson.	5 mai 2021
Billet de blog sur la création d'un graphe de connaissances dans Neptune à l'aide de Data Lens	Consultez le billet de blog Build a knowledge graph in Amazon Neptune using Data Lens de Russell Waterson.	5 mai 2021

Les versions 1.0.1.0, 1.0.1.1 et 1.0.1.2 du moteur sont désormais obsolètes	À partir de maintenant, aucune nouvelle instance de base de données ne sera créée à l'aide de l'une de ces versions de moteur ou de correctifs associés.	26 avril 2021
Traduction en anglais d'une étude de cas sur le Ministère japonais de l'Économie, du Commerce et de l'Industrie utilisant Neptune	Consultez le billet de blog Japan's Ministry of Economy, Trade and Industry Powers gBizINFO Corporate Information Search Database with AWS.	31 mars 2021
Billet de blog sur l'utilisation de Neptune avec Amazon Comprehend et Lex	Consultez le billet de blog Supercharge your knowledge graph using Amazon Neptune, Amazon Comprehend, and Amazon Lex de Dave Bechberger.	31 mars 2021
Billet de blog sur l'utilisation des fonctions Lambda avec Neptune	Voir Utiliser les fonctions AWS Lambda avec Amazon Neptune, par Ian Robinson.	26 mars 2021
Version de moteur 1.0.4.1.R1.1	Depuis le 22 mars 2021, la version 1.0.4.1.R1.1 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.4.1.R1.1 du moteur Neptune.	22 mars 2021

Version de moteur 1.0.4.1.R2.1	Depuis le 11 mars 2021, la version 1.0.4.1.R2.1 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.4.1.R2.1 du moteur Neptune.	11 mars 2021
Billet de blog sur l'utilisation du bloc-notes orienté graphe open source de Neptune pour la visualisation des graphes	Consultez le billet de blog Getting started with open source graph notebook for graph visualization de Joy Wang, Ora Lassila et Stephen Mallette.	10 mars 2021
Didacticiel sur l'intégration de Neptune dans le moteur de découverte de données et de métadonnées Amundsen	Consultez le billet de blog How to use Amundsen with Amazon Neptune d'Andrew Ciambrone.	2 mars 2021
Version de moteur 1.0.4.1.R2	Depuis le 24 février 2021, la version 1.0.4.1.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.4.1.R2 du moteur Neptune.	24 février 2021

Version de moteur 1.0.4.0.R2	Depuis le 24 février 2021, la version 1.0.4.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.4.0.R2 du moteur Neptune.	24 février 2021
Version de moteur 1.0.3.0.R3	Depuis le 19 février 2021, la version 1.0.3.0.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.3.0.R3 du moteur Neptune.	19 février 2021
Version de moteur 1.0.2.2.R6	Depuis le 19 février 2021, la version 1.0.2.2.R6 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.2.2.R6 du moteur Neptune.	19 février 2021

Billet de blog sur la création d'un graphe de connaissances à l'aide des événements Amazon Comprehend	Consultez le billet de blog Building a knowledge graph in Amazon Neptune using Amazon Comprehend Events de Brian O'Keefe, Graham Horwood et Navtanay Sinha.	19 janvier 2021
Billet de blog sur l'activation des applications de données de graphe à faible code	Consultez le billet de blog Enabling low code graph data apps with Amazon Neptune and Graphistry de Leo Meyerovich, Dave Bechberger et Taylor Riggan.	18 janvier 2021
Ajout de la documentation de bloc-notes pour démarrer avec les données de graphe.	Ajout d'une section intégrant le workbench Neptune qui permet de commencer à créer des données de graphe et à développer des applications de graphe sans avoir à créer un cluster Neptune au préalable.	15 janvier 2021
Billet de blog sur la réinitialisation des données d'un graphe Neptune en quelques secondes	Consultez le billet de blog Resetting your graph data in Amazon Neptune in seconds de Niraj Jetly et Navtanay Sinha.	17 décembre 2020
Article de blog sur la façon dont Novartis AG utilise et SageMaker Neptune avec BERT	See Novartis AG utilise Amazon SageMaker et Amazon Neptune pour créer et enrichir un graphe de connaissances à l'aide du BERT, par Othmane Hamzaoui, Fatema Alkhanaizi et Viktor Malesevic.	14 décembre 2020

Billet de blog sur la création d'un graphe de connaissances à l'aide de réseaux thématiques	Consultez le billet de blog Building a knowledge graph with topic networks in Amazon Neptune d'Edward Brown, responsable des projets d'intelligence artificielle, Eduardo Piairo, architecte, Marcia Oliveira, spécialiste des données en chef, et Jack Hampson, PDG de Deeper Insights.	14 décembre 2020
Version de moteur 1.0.4.1	Depuis le 8 décembre 2020, la version 1.0.4.1 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.4.1 du moteur Neptune.	8 décembre 2020
Billet de blog sur la prise en main de Neptune ML	Consultez le billet de blog How to get started with Neptune ML de George Karypis, Dave Bechberger et Karthik Bharathy.	8 décembre 2020
Neptune dispose désormais d'une API de réinitialisation rapide	Grâce à l'API de réinitialisation rapide, vous pouvez supprimer rapidement et facilement toutes les données d'un cluster de bases de données. Consultez API de réinitialisation rapide.	4 décembre 2020

Billet de blog sur la création d'un graphe de connaissances biologiques chez Pendulum	Consultez le billet de blog Building a biological knowledge graph at Pendulum using Amazon Neptune de Connor Skennerton.	26 novembre 2020
Article de blog sur les nouvelles fonctionnalités de Neptune TinkerPop 3.4.8	Découvrez les nouvelles fonctionnalités d'Apache TinkerPop 3.4.8 dans Amazon Neptune, par Stephen Mallette.	18 novembre 2020
Billet de blog sur l'utilisation du service de recherche Amazon Kendra avec Neptune	Consultez le billet de blog Incorporating your enterprise knowledge graph into Amazon Kendra de Yazdan Shirvany, Mohit Mehta et Dipto Chakravarty.	17 novembre 2020
Notifications d'événements désormais disponibles	Neptune prend désormais en charge les notifications d'événements que vous pouvez utiliser pour surveiller plus facilement les clusters de bases de données. Consultez Utilisation de la notification d'événement Neptune.	29 octobre 2020
Points de terminaison personnalisés désormais disponibles	Neptune prend désormais en charge les points de terminaison personnalisés pour un meilleur contrôle de la connexion aux instances de base de données. Consultez Connexion aux points de terminaison Amazon Neptune.	29 octobre 2020

Article de blog sur l'utilisation du AWS Database Migration Service (DMS) pour remplir votre graphe Neptune	Consultez la section Remplissage de votre graphe dans Amazon Neptune à partir d'une base de données relationnelle à l' AWS aide du Database Migration Service (DMS) — Partie 4 : Réunir les deux, par Chris Smith.	22 octobre 2020
Article de blog sur l'utilisation du AWS Database Migration Service (DMS) pour remplir votre graphe Neptune	Consultez la section Remplissage de votre graphique dans Amazon Neptune à partir d'une base AWS de données relationnelle à l'aide du Database Migration Service (DMS) — Partie 3 : Conception du modèle RDF, par Chris Smith.	22 octobre 2020
Article de blog sur l'utilisation du AWS Database Migration Service (DMS) pour remplir votre graphe Neptune	Consultez la section Remplissage de votre graphe dans Amazon Neptune à partir d'une base AWS de données relationnelle à l'aide du Database Migration Service (DMS) — Partie 2 : Conception du modèle de graphe de propriétés, par Chris Smith.	22 octobre 2020

Article de blog sur l'utilisation du AWS Database Migration Service (DMS) pour remplir votre graphe Neptune	Consultez la section Remplissage de votre graphique dans Amazon Neptune à partir d'une base de données relationnelle à l' AWS aide du Database Migration Service (DMS) — Partie 1 : Préparer le terrain, par Chris Smith.	22 octobre 2020
Version de moteur 1.0.4.0	Depuis le 12 octobre 2020, la version 1.0.4.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.4.0 du moteur Neptune.	12 octobre 2020
Version de moteur 1.0.3.0.R2	Depuis le 12 octobre 2020, la version 1.0.3.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.3.0.R2 du moteur Neptune.	12 octobre 2020

Version de moteur 1.0.2.2.R5	Depuis le 12 octobre 2020, la version 1.0.2.2.R5 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.2.2.R5 du moteur Neptune.	12 octobre 2020
Billet de blog sur la configuration du VPC pour les requêtes fédérées SPARQL	Consultez le billet de blog Configure Amazon VPC for SPARQL 1.1 Federated Query with Amazon Neptune de Charles Ivie.	12 octobre 2020
Billet de blog sur l'écriture d'une suppression en cascade en SPARQL	Consultez le billet de blog Write a cascading delete in SPARQL d'Ora Lassila.	5 octobre 2020
Article de blog sur la représentation graphique des AWS ressources à l'aide de Neptune	Consultez la section Représentation graphique de vos AWS ressources avec Amazon Neptune, par Dave Bechberger.	28 septembre 2020
Billet de blog sur l'élaboration d'un mappage terminologique MedDRA pour la pharmacovigilance et le signalement des effets indésirables à l'aide de Neptune	Consultez le billet de blog Building Amazon Neptune based MedDRA terminology mapping for pharmacovigilance and adverse event reporting de Vaijayanti Joshi, Deven Atnoor, Ph.D. et Sudhanshu Malhotra.	24 septembre 2020

Billet de blog sur la création d'un graphe de connaissances à partir d'un entrepôt de données à l'aide de Neptune, pour compléter l'intelligence commerciale	Consultez le billet de blog Complement Commercial Intelligence by Building a Knowledge Graph out of a Data Warehouse with Amazon Neptune de Shahria Hossain et Mikael Graindorge.	23 septembre 2020
Billet de blog sur l'équilibrage de charge à l'aide du client Neptune Gremlin	Consultez le billet de blog Load balance graph queries using the Amazon Neptune Gremlin Client d'Ian Robinson.	16 septembre 2020
Billet de blog sur la personnalisation numérique à l'aide d'un graphe d'identité chez Cox Automotive	Consultez le billet de blog Cox Automotive scales digital personalization using an identity graph powered by Amazon Neptune de Carlos Rendon et Niraj Jetly.	16 septembre 2020
Billet de blog sur le filtrage collaboratif des données Yelp	Consultez le billet de blog Using collaborative filtering on Yelp data to build a recommendation system in Amazon Neptune de Chad Tindel.	8 septembre 2020
Billet de blog sur la visualisation des résultats des requêtes dans Amazon Neptune	Consultez le billet de blog Visualize query results using the Amazon Neptune workbench de Kelvin Lawrence.	2 septembre 2020

Neptune a publié une visualisation de graphe	Amazon Neptune fournit désormais des fonctionnalités étendues de visualisation orientée graphe dans les blocs-notes Jupyter dans le workbench Neptune, ainsi que plusieurs nouvelles fonctionnalités qui facilitent l'utilisation des blocs-notes. Consultez Graph visualization (Visualisation orientée graphe).	12 août 2020
Lancement de Neptune dans la région Amérique du Sud (São Paulo)	Amazon Neptune est désormais disponible dans la région Amérique du Sud (Sao Paulo) (`sa-east-1`).	6 août 2020
Lancement de Neptune dans la région Asie-Pacifique (Hong Kong)	Amazon Neptune est désormais disponible dans la région Asie-Pacifique (Hong Kong) (`ap-east-1`).	6 août 2020
Version de moteur 1.0.3.0	Depuis le 3 août 2020, la version 1.0.3.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.3.0 du moteur Neptune.	3 août 2020

Version de moteur 1.0.2.2.R4	Depuis le 23 juillet 2020, la version 1.0.2.2.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.2.2.R4 du moteur Neptune.	23 juillet 2020
Billet de blog sur le suivi automatique des contacts de Zerobase à l'aide d'Amazon Neptune	Consultez le billet de blog Zerobase creates private, secure, and automated contact tracing using Amazon Neptune de David Harris et Aron Szanto.	13 juillet 2024
Lancement de Neptune dans la région USA Ouest (Californie du Nord)	Amazon Neptune est désormais disponible dans la région USA Ouest (Californie du Nord) (`us-west-1`).	9 juillet 2020
Amazon Neptune prend en charge le contrôle d'accès basé sur les balises	Vous pouvez désormais utiliser des AWS balises dans les politiques IAM pour contrôler l'accès à votre base de données Neptune. Consultez Contrôle d'accès basé sur les balises dans Amazon Neptune.	7 juillet 2020

Un interrogateur de flux Java est désormais disponible	Amazon Neptune prend désormais en charge une version Java de l'interro gateur de flux Lambda pour les flux Neptune ainsi que la version Python. Consultez Ajout de détails sur la pile de consommateurs de flux Neptune que vous créez.	6 juillet 2020
Article de blog sur le graphe de connaissances sur AWS la COVID-19	Voir Création et interroga tion du graphe de connaissa nces sur la AWS COVID-19, par Ninad Kulkarni, Colby Wise, George Price et Miguel Romero.	1er juillet 2020
Version de moteur 1.0.1.1	Depuis le 26 juin 2020, la version 1.0.1.1 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.1.1 du moteur Neptune.	26 juin 2020
Billet de blog sur la migration de Blazegraph vers Amazon Neptune	Consultez le billet de blog Moving to the cloud: Migrating Blazegraph to Amazon Neptune de Dave Bechberger.	25 juin 2020

Billet de blog sur la technologie Change Data Capture de Neo4j à Amazon Neptune	Consultez le billet de blog Change data capture from Neo4j to Amazon Neptune using Amazon Managed Streaming for Apache Kafka de Sanjeet Sahay.	22 juin 2020
Billet de blog sur la façon dont Waves utilise Amazon Neptune	Consultez Comment Waves exécute les requêtes et recommandations des utilisateurs à grande échelle avec Amazon Neptune de Pavel Vasilyev.	16 juin 2020
Version de moteur 1.0.1.2	Depuis le 10 juin 2020, la version 1.0.1.2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.1.2 du moteur Neptune.	10 juin 2020
Billet de blog sur la création d'un référentiel de connaissances clients	Consultez Création d'un référentiel global de connaissances clients avec Amazon Neptune et Amazon Redshift de Ram Bhandarkar.	9 juin 2020
Billet de blog sur la façon dont Gunosy utilise Amazon Neptune	Consultez Comment Gunosy a créé une fonctionnalité de commentaire dans News Pass en utilisant Amazon Neptune de Yosuke Uchiyama.	8 juin 2020

Article de blog sur le graphe de connaissances sur AWS la COVID-19	Voir Création et interrogation du graphe de connaissances sur la AWS COVID-19 par Ninad Kulkarni, Colby Wise, George Price et Miguel Romero.	2 juin 2020
Billet de blog sur l'exploration de la recherche COVID-19 à l'aide d'Amazon Neptune	Consultez le billet de blog Exploring scientific research on COVID-19 with Amazon Neptune, Amazon Comprehend Medical, and the Tom Sawyer Graph Database Browser de George Price, Colby Wise, Miguel Romero et Ninad Kulkarni.	2 juin 2020
Vous pouvez désormais charger des données dans Neptune en utilisant AWS DMS	Consultez la section Utilisation du service AWS de migration de base de données pour charger des données dans Amazon Neptune à partir d'un autre magasin de données.	1er juin 2020
La version 1.0.2.0 du moteur devient obsolète	La version 1.0.2.0 du moteur Amazon Neptune est désormais obsolète. Les clusters qui s'exécutent sur cette version de moteur seront automatiquement mis à niveau vers la version 1.0.2.1 pendant la première fenêtre de maintenance suivant le 1er juin 2020.	19 mai 2020

Billet de blog sur la création d'un graphe d'identité client à l'aide de Neptune	Consultez le billet de blog Building a customer identity graph with Amazon Neptune de Rajesh Wunnava et Taylor Riggan.	12 mai 2020
Version de moteur 1.0.2.0.R3	Depuis le 5 mai 2020, la version 1.0.2.0.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.2.0.R3 du moteur Neptune.	5 mai 2020
Version de moteur 1.0.2.1.R6	Depuis le 22 avril 2020, la version 1.0.2.1.R6 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.2.1.R6 du moteur Neptune.	22 avril 2020
Billet de blog sur la migration des données de Neo4j vers Neptune	Consultez Migrating a Neo4j graph database to Amazon Neptune with a fully automated utility, de Sanjeet Sahay.	13 avril 2020

Billet de blog sur la réduction du coût de construction d'applications orientées graphe avec Neptune	Consultez Lower the cost of building graph apps by up to 76% with Amazon Neptune T3 instances, de Karthik Bharathy et Brad Bebee.	9 avril 2020
Neptune offre une classe d'instances extensible T3	Vous pouvez désormais créer une instance extensible Amazon Neptune T3 à des fins de développement et de test à moindres coûts. Consultez Classe d'instances extensible T3 Neptune.	8 avril 2020
Version de moteur 1.0.2.2.R2	Depuis le 2 avril 2020, la version 1.0.2.2.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.2.2.R2 du moteur Neptune.	2 avril 2020
Billet de blog sur la création de graphe de dépendance des investissements chez EDGAR	Consultez Graphing investmen t dependency with Amazon Neptune de Lawrence Verdi.	17 mars 2020
Lancement de Neptune dans la région Europe (Paris)	Amazon Neptune est désormais disponible dans la région Europe (Paris) (eu-west-3).	11 mars 2020

Version de moteur 1.0.2.2	Depuis le 9 mars 2020, la version 1.0.2.2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponibl e dans chaque région. Pour plus d'informations sur cette version du moteur, consultez la section Version 1.0.2.2 du moteur Neptune.	9 mars 2020
Arrêt et redémarrage d'un cluster de bases de données	Vous pouvez désormais arrêter un cluster de bases de données pendant 7 jours à l'aide de la console Neptune, puis le redémarrer lorsque vous en avez à nouveau besoin. Pendant que votre cluster de bases de données est à l'arrêt, vous êtes facturé uniquement pour le stockage du cluster, les instantan és manuels et le stockage des sauvegardes automatiq ues, mais pas pour les heures d'instance de base de données. Consultez la section Arrêt et démarrage d'un cluster de bases de données Amazon Neptune.	19 février 2020

Vidéo sur un graphe social chez Nike	Écoutez Todd Escalona AWS parler avec Marc Wangenheim, directeur technique senior chez Nike, de la façon dont l'entreprise gère un certain nombre d'applications via un graphe social construit sur Amazon Neptune. Consultez Nike: A Social Graph at Scale with Amazon Neptune.	11 février 2020

Les clusters Neptune peuvent désormais être configurés pour exiger des connexions SSL	Dans les régions qui prennent toujours en charge les connexions HTTP, SSL est désormais activé par défaut dans tous les nouveaux groupes de paramètres. Aucune modification n'est apportée aux groupes de paramètres existants, mais vous pouvez forcer les clients à utiliser SSL en affectant la valeur 1 au paramètre `neptune_enforce_ssl` . Consultez les informations relatives au chiffrement en transit et à la connexion à Neptune à l'aide de SSL/HTTPS pour plus de détails sur l'activation des connexions HTTP pour un cluster dans une région qui les prend toujours en charge. Consultez les informations relatives aux paramètres que vous pouvez utiliser pour configurer Amazon Neptune pour obtenir une description des paramètres de cluster et d'instance.	10 février 2020

Vous pouvez désormais spécifier la version du moteur et la protection contre les suppressions dans le modèle de CloudFormation Neptune	Amazon Neptune a mis à jour son CloudFormation modèle pour inclure un `AWS::Neptune::DBCluster.EngineVersion` paramètre qui vous permet de spécifier une version de moteur particulière pour votre nouveau cluster de bases de données, ainsi qu'un `AWS::Neptune::DBCluster.DeletionProtection` paramètre qui vous permet d'activer la protection contre la suppression pour celui-ci.	9 février 2020
Deletion protection (Protection contre la suppression)	Amazon Neptune fournit une protection contre la suppression pour les clusters et les instances de base de données. Tant que la protection contre la suppression est activée sur un cluster ou une instance de base de données, vous ne pouvez pas supprimer cet élément. Consultez Impossible de supprimer une instance de base de données si la protection contre la suppression est activée.	20 janvier 2020
Lancement de Neptune dans la région Chine (Ningxia)	Amazon Neptune est désormais disponible dans la région Chine (Ningxia) (cn-northwest-1).	15 janvier 2020

Version de moteur 1.0.2.1.R4	Le correctif R4 pour la version de moteur 1.0.2.1 est disponible globalement. Pour plus d'informations, consultez la section Version 1.0.2.1.R4 du moteur Neptune.	20 décembre 2019
Version de moteur 1.0.2.1.R3	Le correctif R3 pour la version de moteur 1.0.2.1 est disponible globalement. Pour plus d'informations, consultez la section Version 1.0.2.1.R3 du moteur Neptune.	12 décembre 2019
Billet de blog sur l'utilisation de Neptune pour analyser les flux de réseaux sociaux	Consultez Analyse des flux de réseaux sociaux à l'aide d'Amazon Neptune.	27 novembre 2019
Version de moteur 1.0.2.1.R2	Le correctif R2 pour la version de moteur 1.0.2.1 est disponible globalement. Pour plus d'informations, consultez la section Version 1.0.2.1.R2 du moteur Neptune.	25 novembre 2019
Version de moteur 1.0.2.1.R1	La version 1.0.2.1.R1 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la section Version 1.0.2.1 du moteur Neptune.	22 novembre 2019

Version de moteur 1.0.2.0.R2	Le correctif R2 de la version de moteur 1.0.2.0 est disponible globalement. Pour plus d'informations, consultez la section Version 1.0.2.0.R2 du moteur Neptune.	21 novembre 2019
Billet de blog sur les sessions et ateliers Neptune à re:Invent 2019	Consultez votre guide des sessions, des ateliers et des conférences à la craie Amazon Neptune à AWS l'occasion de re:Invent 2019.	20 novembre 2019
Version de moteur 1.0.2.0.R1	La version 1.0.2.0.R1 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la section Version 1.0.2.0 du moteur Neptune.	8 novembre 2019
Billet de blog sur la capture des modifications de graphe à l'aide de Neptune Streams	Consultez Capture Graph Changes using Neptune Streams.	6 novembre 2019
Version de moteur 1.0.1.0.200502.0	La version 1.0.1.0.200502.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200502.0.	31 octobre 2019
Lancement de Neptune dans la région Moyen-Orient (Bahreïn)	Amazon Neptune est désormais disponible dans la région Moyen-Orient (Bahreïn) (me-south-1).	30 octobre 2019

Lancement de Neptune dans la région Canada (Centre)	Amazon Neptune est désormais disponible dans la région Canada (Centre) (ca-central-1).	30 octobre 2019
Billet de blog sur la nouvelle fonctionnalité SPARQL Streams de Neptune et la prise en charge des requêtes fédérées SPARQL	Consultez Amazon Neptune releases Streams, SPARQL federated query for graphs and more.	17 octobre 2019
Version de moteur 1.0.1.0.200463.0	La version 1.0.1.0.200463.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200463.0.	15 octobre 2019
Version de moteur 1.0.1.0.200457.0	La version 1.0.1.0.200457.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200457.0.	19 septembre 2019
Billet de blog sur la nouvelle fonctionnalité SPARQL explain de Neptune	Lisez Using SPARQL explain to understand query execution in Amazon Neptune.	17 septembre 2019
Article de blog sur le support de Neptune pour 3.4 TinkerPop	Découvrez qu'Amazon Neptune prend désormais en charge les fonctionnalités TinkerPop 3.4.	6 septembre 2019

Article de blog sur l'utilisation de Neptune sur Amazon PyTorch SageMaker	Consultez la section Une expérience personnalisée shop-by-style « » d'utilisation PyTorch sur Amazon SageMaker et Amazon Neptune.	22 août 2019
Article de blog sur l'utilisation de Neptune avec AWS AppSync Amazon Elasticache	Consultez la section Intégration de sources de données alternatives avec AWS AppSync : Amazon Neptune et Amazon. ElastiCache	22 août 2019
Neptune a été lancé en AWS GovCloud (Est des États-Unis)	Amazon Neptune est désormais disponible en AWS GovCloud (est des États-Unis) (-1). us-gov-east	21 août 2019
Neptune a été lancé en AWS GovCloud (ouest des États-Unis)	Amazon Neptune est désormais disponible en AWS GovCloud (ouest des États-Unis) (-1). us-gov-west	14 août 2019
Version de moteur 1.0.1.0.200369.0	La version 1.0.1.0.200369.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200369.0.	13 août 2019
Version de moteur 1.0.1.0.200366.0	La version 1.0.1.0.200366.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200366.0.	26 juillet 2019

Article de blog sur l'utilisation de Neptune sur Amazon PyTorch SageMaker	Consultez la section Une expérience personnalisée shop-by-style « » d'utilisation PyTorch sur Amazon SageMaker et Amazon Neptune.	3 juillet 2019
Version de moteur 1.0.1.0.200348.0	La version 1.0.1.0.200348.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200348.0.	2 juillet 2019
Lancement de Neptune dans la région Europe (Stockholm)	Amazon Neptune est désormais disponible dans la région Europe (Stockholm) (eu-north-1).	27 juin 2019
Neptune peut désormais publier des journaux d'audit dans Logs CloudWatch	Pour plus d'informations, consultez la section Publication de Neptune Logs sur Amazon CloudWatch Logs.	18 juin 2019
Version de moteur 1.0.1.0.200310.0	La version 1.0.1.0.200310.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200310.0.	12 juin 2019
Article de blog à propos LifeOmic de JupiterOne	Découvrez LifeOmicHow's JupiterOne simplifie les opérations de sécurité et de conformité avec Amazon Neptune.	2 mai 2019

Lancement de Neptune dans la région Asie-Pacifique (Séoul)	Amazon Neptune est désormais disponible dans la région Asie-Pacifique (Séoul) (ap-northeast-2).	1er mai 2019
Version de moteur 1.0.1.0.200296.0	La version 1.0.1.0.200296.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200296.0.	1er mai 2019
Lancement de Neptune dans la région Asie-Pacifique (Mumbai)	Amazon Neptune est désormais disponible dans la région Asie-Pacifique (Mumbai) (ap-south-1).	6 mars 2019
Billet de blog relatif aux indicateurs de requête Gremlin	Consultez la page relative à la présentation des indicateurs de requête Gremlin pour Amazon Neptune.	26 février 2019
Lancement de Neptune dans la région Asie-Pacifique (Tokyo)	Amazon Neptune est désormais disponible dans la région Asie-Pacifique (Tokyo) (ap-northeast-1).	23 janvier 2019
CloudFormation modèle pour créer une AWS Lambda fonction pour accéder à Neptune	Mise à jour de la section de démarrage et ajout d'un CloudFormation modèle pour créer une fonction Lambda à utiliser avec Neptune. Pour en savoir plus, consultez Mise en route avec Neptune.	23 janvier 2019

Version de moteur 1.0.1.0.200267.0	La version 1.0.1.0.200267.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200267.0.	21 janvier 2019
Lancement de Neptune dans la région Asie-Pacifique (Sydney)	Amazon Neptune est désormais disponible dans la région Asie-Pacifique (Sydney) (ap-southeast-2).	9 janvier 2019
Billet de blog relatif à l'utilisation de Metaphactory	Consultez la page relative à l'exploration de graphes de connaissances sur Amazon Neptune à l'aide de Metaphactory.	9 janvier 2019
Lancement de Neptune dans la région Asie-Pacifique (Singapour)	Amazon Neptune est désormais disponible dans la région Asie-Pacifique (Singapour) (ap-southeast-1).	13 décembre 2018
Version de moteur 1.0.1.0.200264.0	La version 1.0.1.0.200264.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200264.0.	19 novembre 2018
Prise en charge SSL d'Amazon Neptune	Neptune prend désormais en charge les connexions SSL.	19 novembre 2018
Rubriques d'erreur consolidées	Toutes les informations sur les messages et les codes d'erreur sont désormais regroupées dans une seule rubrique.	15 novembre 2018

Mise à jour de la rubrique de mise en route	Mise à jour de la rubrique de mise en route avec des liens supplémentaires et une documentation réorganisée.	14 novembre 2018
Version de moteur 1.0.1.0.200258.0	La version 1.0.1.0.200258.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200258.0.	le 8 novembre 2018
Lancement de Neptune dans la région Europe (Francfort)	Amazon Neptune est désormais disponible dans la région Europe (Francfort) (eu-central-1).	7 novembre 2018
Premier billet de blog de la série	Consultez la page relative à la création de graphe (Partie 1 : voies aériennes).	7 novembre 2018
Article de blog sur l'utilisation des blocs-notes Amazon SageMaker Jupyter	Consultez Analyser les graphiques Amazon Neptune à l'aide des blocs-notes Amazon SageMaker Jupyter.	1er novembre 2018
Version de moteur 1.0.1.0.200255.0	La version 1.0.1.0.200255.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200255.0.	29 octobre 2018
Lancement de Neptune dans la région Europe (Londres)	Amazon Neptune est désormais disponible dans la région Europe (Londres) (eu-west-2).	3 octobre 2018

Version de moteur 1.0.1.0.200237.0	La version 1.0.1.0.200237.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200237.0.	6 septembre 2018
Version de moteur 1.0.1.0.200236.0	La version 1.0.1.0.200236.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200236.0.	24 juillet 2018
Version de moteur 1.0.1.0.200233.0	La version 1.0.1.0.200233.0 du moteur Amazon Neptune est disponible globalement. Pour plus d'informations, consultez la mise à jour 1.0.1.0.200233.0.	22 juin 2018
Nouveau guide de démarrage rapide Neptune	Démarrage rapide mis à jour avec CloudFormation le didacticiel de la console G705. Pour plus d'informations, consultez Amazon Neptune Quick Start Using. CloudFormation	19 juin 2018
Version initiale d'Amazon Neptune	Il s'agit de la première version du Guide de l'utilisateur Neptune. Consultez également le billet de blog de cette version relatif à Amazon Neptune disponible pour tous.	30 mai 2018

Billet de blog d'introduction à Neptune	Consultez la page relative à Amazon Neptune, un service de bases de données orientées graphe entièrement géré.	29 novembre 2017

Premiers pas avec Amazon Neptune

Amazon Neptune est un service de base de données orientée graphe entièrement géré qui évolue pour gérer des milliards de relations et vous permet de les interroger avec une latence de quelques millisecondes, à un faible coût pour ce type de capacité.

Si vous recherchez des informations plus détaillées sur Neptune, consultez Présentation des fonctionnalités Amazon Neptune.

Si vous connaissez déjà les graphiques, passez directement à Démarrage rapide en utilisant CloudShell ouUtilisation de Neptune avec des carnets graphiques. Ou, si vous souhaitez créer immédiatement une base de données Neptune, consultez Création d'un cluster Amazon Neptune à l'aide de AWS CloudFormation.

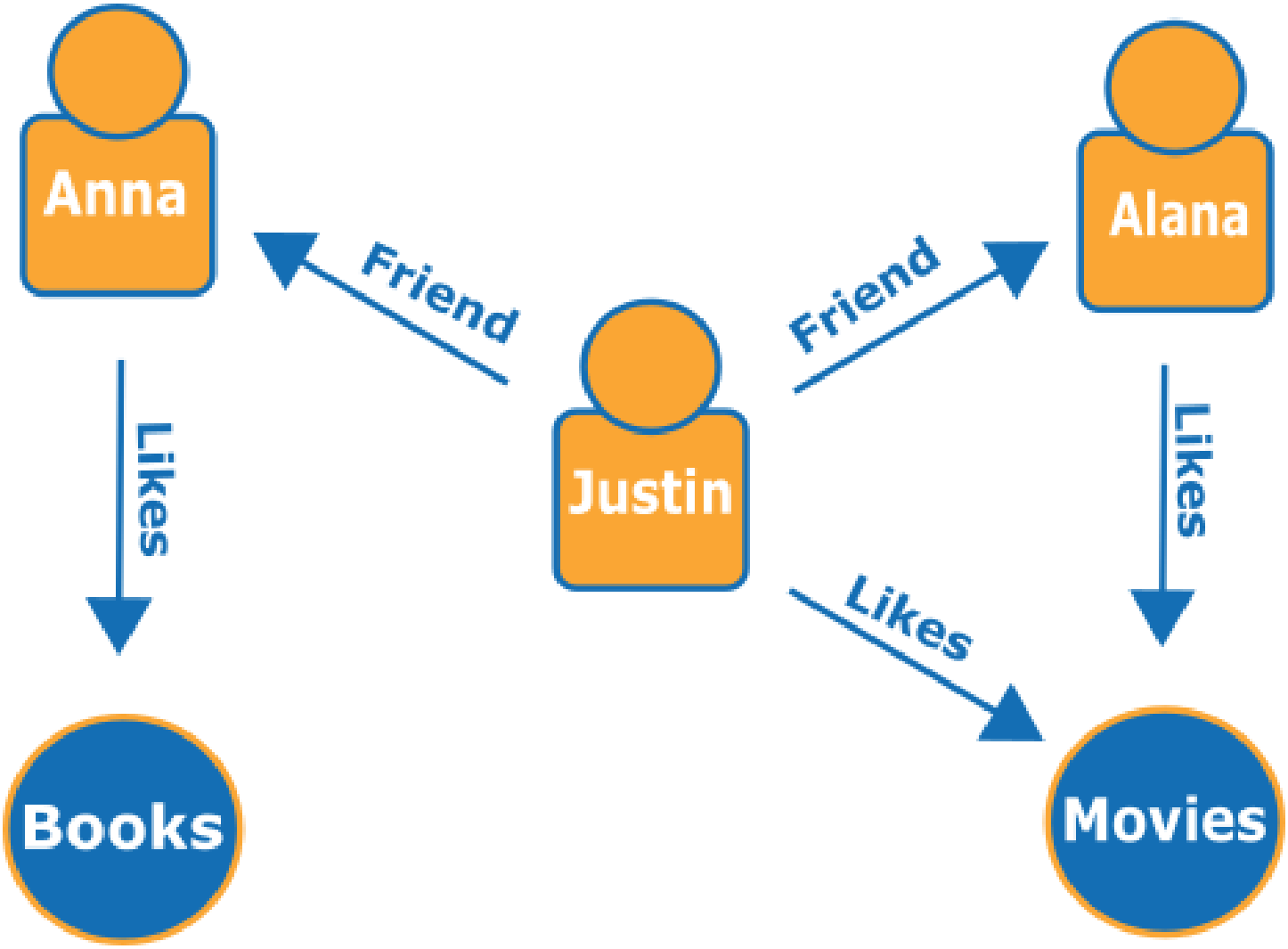
Sinon, vous souhaiterez peut-être en savoir un peu plus sur les bases de données de graphes avant de commencer.

Concepts clés des bases de données graphiques

Les bases de données orientée graphe sont optimisées pour stocker et interroger les relations entre des éléments de données.

Elles stockent les éléments de données sous forme de sommets du graphe, et les relations entre eux sous forme d'arêtes. Chaque arête a un type spécifique et se dirige d'un sommet (le début) à un autre (la fin). Les relations peuvent être appelées prédicats ou arêtes, tandis que les sommets sont également parfois appelés nœuds. Dans ce que l'on appelle les graphes de propriétés, les sommets et les arêtes peuvent également être associés à des propriétés supplémentaires.

Voici un petit graphe représentant les amis et les loisirs d'un réseau social :



Les arêtes sont représentées par des flèches nommées, et les sommets représentent des personnes et des loisirs spécifiques qu'ils relient.

Un parcours simple de ce graphique peut vous renseigner sur ce qu'aiment les amis de Justin.

Pourquoi utiliser une base de données orientée graphe ?

Dès lors que les connexions ou les relations entre les entités constituent le cœur des données que vous essayez de modéliser, le choix doit naturellement se porter sur une base de données orientée graphe.

D'une part, il est facile de modéliser les interconnexions de données sous forme de graphe, puis d'écrire des requêtes complexes qui extraient des informations réelles du graphe.

Pour créer une application équivalente à l'aide d'une base de données relationnelle, vous devez créer de nombreuses tables avec plusieurs clés étrangères, puis écrire des requêtes SQL imbriquées et des jointures complexes. Non seulement cette approche gagne rapidement en complexité du point de vue du codage, mais ses performances se dégradent rapidement à mesure que la quantité de données augmente.

En revanche, une base de données orientée graphe telle que Neptune peut interroger les relations entre des milliards de sommets sans perte de performances.

Que peut-on faire avec une base de données orientée graphe ?

Les graphes peuvent représenter les interrelations entre des entités du monde réel de nombreuses manières, en termes d'actions, de propriété, de filiation, de choix d'achat, de liens personnels, de liens familiaux, etc.

Voici quelques-uns des domaines les plus courants dans lesquels les bases de données orientées graphe sont utilisées :

- Graphes de connaissances : les graphes de connaissances vous permettent d'organiser et d'interroger toutes sortes d'informations connectées pour répondre à des questions générales. À l'aide d'un graphe de connaissances, vous pouvez ajouter des informations thématiques à des catalogues de produits et modéliser diverses informations telles que celles qui se trouvent dans Wikidata.

  Pour en savoir plus sur le fonctionnement des graphes de connaissances et sur leurs champs d'application, consultez la section Graphes de connaissances sur AWS.
- Graphes d'identité : dans une base de données orientée graphe, vous pouvez enregistrer les relations entre des catégories d'informations telles que les centres d'intérêt des clients, les amis et l'historique des achats, puis interroger ces données pour formuler des recommandations personnalisées et pertinentes.

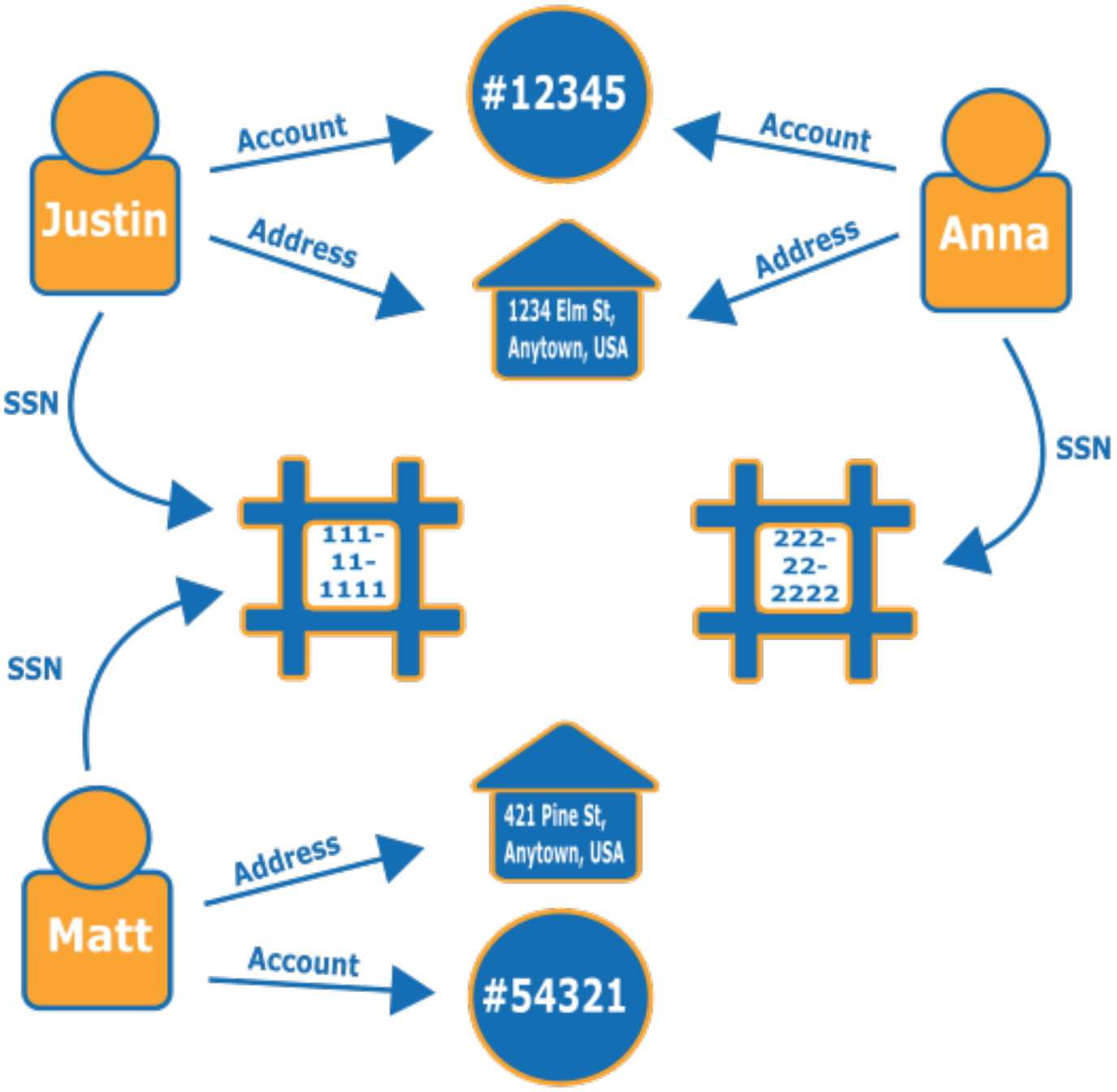
  Par exemple, vous pouvez utiliser une base de données orientée graphe pour recommander des produits à un utilisateur en fonction de ceux achetés par les autres utilisateurs adeptes du même sport et dont l'historique d'achat est similaire. Vous pouvez aussi identifier les personnes ayant un ami en commun, mais qui ne se connaissent pas encore, et émettre une recommandation de mise en relation.

  Les graphes de ce type sont connus sous le nom de graphes d'identité et sont largement utilisés pour personnaliser les interactions avec les utilisateurs. Pour en savoir plus, consultez Graphes

d'identité sur AWS. Pour créer votre propre graphe d'identité, vous pouvez commencer par l'exemple Graphe d'identité avec Amazon Neptune.

- Graphes de détection de fraude : il s'agit d'une utilisation courante des bases de données orientées graphe. Ces graphes peuvent vous aider à suivre les achats par carte de crédit et les points de vente afin de détecter toute utilisation inhabituelle ou de détecter les situations où un acheteur essaie d'utiliser la même adresse e-mail et la même carte de crédit que celles utilisées dans un cas de fraude connu. Ils permettent de rechercher plusieurs personnes associées à une adresse e-mail privée ou plusieurs personnes partageant la même adresse IP dans différents lieux physiques.

  Prenons le graphe suivant. Il illustre la relation entre trois personnes et les informations relatives à leur identité. Chaque personne possède une adresse, un compte bancaire et un numéro de sécurité sociale. Cependant, nous pouvons voir que Matt et Justin partagent le même numéro de sécurité sociale, ce qui est interdit et indique une fraude possible de l'un d'eux. Une requête sur un graphe de détection de fraude peut révéler les liens de ce type afin qu'ils puissent être examinés.



Pour en savoir plus sur les graphes de détection de fraude et sur leurs champs d'applications, consultez Graphes de détection de fraude sur AWS.

- Réseaux sociaux : les applications de réseaux sociaux sont l'un des champs d'application principaux et les plus courants des bases de données orientées graphe.

Par exemple, supposons que vous souhaitiez intégrer un flux social à un site web. Vous pouvez facilement utiliser une base de données orientées graphe sur le back-end pour fournir aux utilisateurs des résultats qui reflètent les dernières mises à jour publiées par leur famille, leurs amis, les personnes dont ils « aiment » les mises à jour et les personnes vivant à proximité.

- Indications routières : un graphe peut aider à trouver le meilleur itinéraire entre un point de départ et une destination, compte tenu du trafic actuel et des modèles de trafic habituels.
- Logistique : les graphes peuvent aider à identifier le moyen le plus efficace d'utiliser les ressources de livraison et de distribution disponibles pour répondre aux exigences des clients.
- Diagnostic : les graphes peuvent représenter des arbres de diagnostic complexes qui peuvent être interrogés pour identifier la source des problèmes et des défaillances observés.
- Recherche scientifique : avec une base de données orientée graphe, vous pouvez créer des applications qui stockent et consultent des données scientifiques, voire des informations médicales sensibles, à l'aide du chiffrement au repos. Par exemple, vous pouvez stocker des modèles des maladies et d'interactions géniques. Vous pouvez rechercher des modèles de graphiques dans les interactions protéiques afin de trouver d’autres gènes susceptibles d’être associés à une maladie. Vous pouvez modéliser les composés chimiques sous forme de graphe et les interroger afin d'identifier les redondances dans les structures moléculaires. Vous pouvez corréler les données des patients à partir des dossiers médicaux de différents systèmes. Vous pouvez organiser par thèmes les publications de recherche afin de trouver rapidement des informations pertinentes.
- Exigences réglementaires : vous pouvez stocker des exigences réglementaires complexes sous forme de graphe et les interroger pour identifier les situations où elles pourraient s'appliquer aux activités quotidiennes de votre entreprise.
- Topologie et événements du réseau : une base de données orientée graphe peut vous aider à gérer et à protéger un réseau informatique. Lorsque vous stockez la topologie du réseau sous forme de graphe, vous pouvez également stocker et traiter de nombreux types d'événements sur le réseau. Vous pouvez répondre à des questions telles que le nombre d'hôtes exécutant une application donnée. Vous pouvez y rechercher des modèles susceptibles d'indiquer qu'un hôte spécifique a été compromis par un programme malveillant et identifier les données de connexion permettant de retracer le programme jusqu'à l'hôte d'origine qui l'a téléchargé.

Comment interroge-t-on un graphe ?

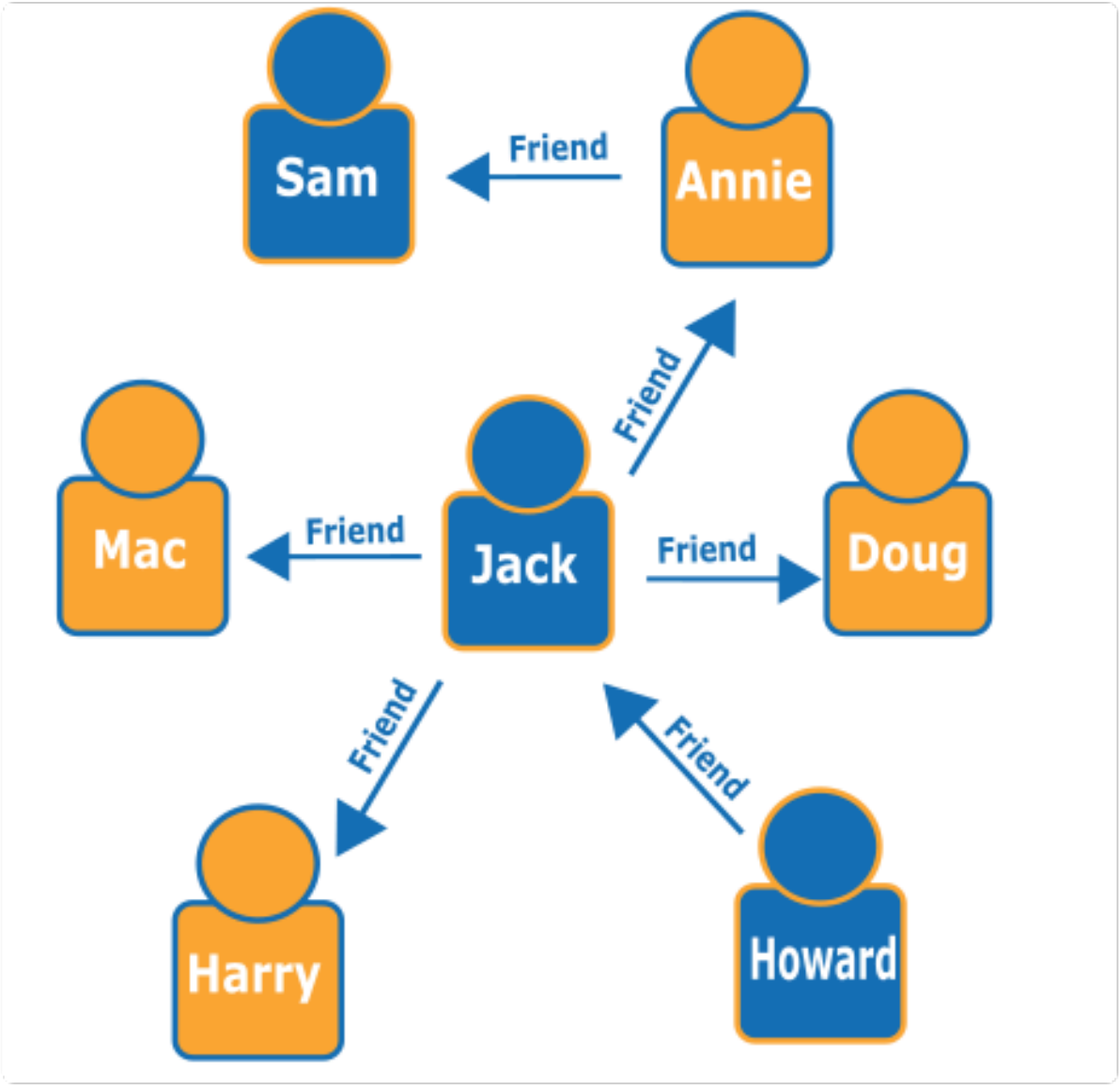
Neptune prend en charge trois langages de requête spécifiques conçus pour interroger des données de graphe de différents types. Vous pouvez utiliser les langages suivants pour ajouter, modifier, supprimer et interroger des données dans une base de données orientée graphe Neptune :

- Gremlin est un langage de parcours de graphe pour les graphes de propriétés. Dans Gremlin, une requête est une traversée composée d'étapes distinctes, chacune suivant une arête jusqu'à un nœud. Consultez la documentation Gremlin sur Apache TinkerPop pour plus d'informations.

  L'implémentation Neptune de Gremlin présente certaines différences par rapport aux autres implémentations, en particulier lorsque vous l'utilisez Gremlin-Groovy (requêtes Gremlin envoyées sous forme de texte sérialisé). Pour de plus amples informations, veuillez consulter Conformité d'Amazon Neptune avec les normes Gremlin.
- openCypher : openCypher est un langage de requête déclaratif pour les graphes de propriétés initialement développé par Neo4j, puis rendu open source en 2015. Il a contribué au projet openCypher sous une licence open source Apache 2. Consultez Cypher Query Language Reference (version 9) pour en savoir plus sur la spécification de ce langage, ainsi que le guide de style Cypher pour plus d'informations.
- SPARQL est un langage de requête déclaratif pour les données RDF. Il repose sur la mise en correspondance de modèles de graphe, qui est normalisée par le World Wide Web Consortium (W3C) et décrite dans la présentation de SPARQL 1.1 ainsi que dans la spécification SPARQL 1.1 Query Language. Consultez Conformité d'Amazon Neptune avec les normes SPARQL pour plus d'informations sur l'implémentation Neptune de SPARQL.

Exemples de correspondance de requêtes Gremlin et SPARQL

Étant donné le graphe suivant de personnes (nœuds) et de leurs relations (arêtes), vous pouvez découvrir qui sont les « amis d'amis » d'une personne donnée, tels que les amis des amis de Howard.



Si vous regardez le graphe, vous pouvez voir que Howard a un ami, Jack, et que Jack a quatre amis : Annie, Harry, Doug et Mac. Il s'agit d'un exemple de graphe simple, mais ces types de requêtes peuvent évoluer en complexité, dimension de l'ensemble de donnés et taille du résultat.

Voici une requête de traversée Gremlin qui renvoie le nom des amis des amis de Howard.

```
g.V().has('name', 'Howard').out('friend').out('friend').values('name')
```

Voici une requête SPARQL qui renvoie le nom des amis des amis de Howard.

```
prefix : <#>

  select ?names where {
    ?howard :name "Howard" .
    ?howard :friend/:friend/:name ?names .
  }
```

> **Note**
>
> Chaque partie d'un triplet RDF (Resource Description Framework) possède un URI qui lui est associé. Dans l'exemple ci-dessus, le préfixe de l'URI est volontairement court.

Suivre un cours en ligne sur l'utilisation d'Amazon Neptune

Si vous aimez apprendre par le biais de vidéos, AWS propose des cours en ligne dans le AWS cadre des Online Tech Talks pour vous aider à démarrer. En voici une qui présente les bases de données orientées graphe :

Présentation, analyse approfondie et démonstration des bases de données orientées graphe avec Amazon Neptune.

Approfondissement de l'architecture de référence des graphes

Lorsque vous réfléchissez aux problèmes qu'une base de données de graphes pourrait résoudre pour vous et à la manière de les aborder, l'un des meilleurs points de départ est le projet d'architectures GitHub de référence de graphes Neptune.

Vous y trouverez des descriptions détaillées des types de charge de travail spécifiques aux graphes, ainsi que trois sections qui vous aideront à concevoir une base de données orientée graphe efficace :

- Modèles de données et langages de requête : cette section explique les différences entre Gremlin et SPARQL et décrit comment choisir entre ces deux langages.
- Modélisation des données de graphe : il s'agit d'une discussion approfondie sur la façon de prendre des décisions en matière de modélisation de données de graphe, y compris des présentations détaillées de la modélisation de graphes de propriétés à l'aide de Gremlin et de la modélisation RDF à l'aide de SPARQL.

- Conversion d'autres modèles de données en modèle de graphe : découvrez ici comment convertir un modèle de données relationnel en modèle orienté graphe.

Trois sections vous indiquent également les étapes spécifiques à suivre pour utiliser Neptune :

- Connexion à Amazon Neptune à partir de clients extérieurs au VPC Neptune : cette section présente plusieurs options de connexion à Neptune à partir de l'extérieur du VPC où se trouve votre cluster de bases de données.
- Accès à Amazon Neptune depuis les fonctions AWS Lambda : vous découvrirez ici comment vous connecter de manière fiable à Neptune à partir des fonctions Lambda.
- Écriture sur Amazon Neptune à partir d'un flux de données Amazon Kinesis : cette section explique comment gérer les scénarios à haut débit d'écriture avec Neptune.

Démarrage rapide en utilisant AWS CloudShell

Ce guide vous aide à démarrer rapidement en vous connectant AWS CloudShellà votre cluster Neptune et en l'interrogeant directement depuis la console de AWS gestion.

Temps de réalisation : environ 2 minutes

Coût : CloudShell est disponible sans frais supplémentaires. La création d'un cluster Neptune entraîne des frais. Pour en savoir plus sur les tarifs, consultez la tarification d'Amazon Neptune.

Conditions préalables

- Un AWS compte. Si vous n'en avez pas, consultez la section Configuration de votre AWS compte.
- Un cluster Neptune dont le statut est Disponible. Si vous n'en avez pas déjà un, reportez-vous Lancement d'un cluster de base de données Neptune à l'aide du AWS Management Console à la section pour en créer un.

Étape 1 : Connectez-vous à votre cluster Neptune

Connectez-vous à votre cluster Neptune à l'aide de. AWS CloudShell CloudShellprovisionne automatiquement un environnement shell connecté au VPC de votre cluster. Il n'est pas nécessaire de configurer des groupes de sécurité ou de configurer des blocs-notes pour interagir avec la base de données.

Pour vous connecter à votre cluster

1. Dans la console Neptune, sous Clusters, localisez le cluster que vous avez créé.
2. Cochez la case à côté de votre cluster.
3. Choisissez Se connecter au cluster.

   > **Note**
   >
   > Ce bouton devient actif une fois que vous avez coché la case à côté de votre cluster et que l'état du cluster s'affiche comme Disponible.

   L'écran CloudShell Exécuter la commande apparaît.
4. Choisissez Exécuter pour vous connecter au cluster. Le provisionnement de l'environnement VPC prend quelques secondes.

Une fois que vous êtes connecté, la `graphsh` console apparaît et fournit un shell interactif pour exécuter des requêtes graphiques.

Étape 2 : Insérer et interroger des données

Maintenant que vous êtes connecté à votre cluster, exécutez quelques requêtes pour vous familiariser avec Amazon Neptune. Les exemples suivants utilisent l'un openCypher des langages de requête pris en charge par Neptune.

Pour insérer des nœuds et des relations

Exécutez la requête suivante pour créer des nœuds pour deux personnes et une relation :

```
CREATE (a:Person {name: 'Alice', age: 30}), (b:Person {name: 'Bob', age: 40}), (a)-
[:KNOWS]->(b)
```

La sortie ressemble à ce qui suit:

```
##########
# result #
##########
# []     #
##########
```

Pour trouver des nœuds

Exécutez la requête suivante pour renvoyer tous les noms de personnes par ordre alphabétique :

```
MATCH (p:Person) RETURN p.name AS name ORDER BY name
```

La sortie ressemble à ce qui suit:

```
#########
# name  #
#########
# Alice #
# Bob   #
#########
```

Étapes suivantes

Vous avez terminé ce démarrage rapide. Pour continuer à apprendre, explorez les fonctionnalités suivantes :

- Présentation des fonctionnalités Amazon Neptune— Découvrez les principales fonctionnalités d'Amazon Neptune.
- Amazon Neptune sans serveur— Adaptez automatiquement la capacité de votre cluster en fonction de la charge de travail.
- Gestion de votre base de données Amazon Neptune— Gérez vos clusters, instances et configurations.

Pour éviter des frais permanents, supprimez le cluster Neptune si vous l'avez créé uniquement pour ce démarrage rapide. Pour obtenir des instructions, veuillez consulter Gestion de votre base de données Amazon Neptune.

Utilisation d'Amazon Neptune avec des carnets graphiques

Pour travailler avec des graphes Neptune, vous pouvez utiliser un bloc-notes Neptune ou créer une nouvelle base de données Neptune à l'aide d'un modèle. CloudFormation

Que vous soyez novice en matière de graphes et que vous souhaitiez apprendre et expérimenter, ou que vous soyez expérimenté et que vous souhaitiez affiner vos requêtes, le pupitre Neptune propose

un environnement de développement interactif (IDE) qui peut améliorer votre productivité lorsque vous créez des applications graphiques. Le Workbench fournit une interface conviviale pour interagir avec votre base de données Neptune, écrire des requêtes et visualiser vos données.

En utilisant le CloudFormation modèle pour configurer votre base de données Neptune et le Workbench pour développer vos applications graphiques, vous pouvez démarrer avec Neptune rapidement et efficacement, sans avoir besoin d'outils supplémentaires. Cela vous permet de vous concentrer sur le développement de vos applications plutôt que sur la configuration de l'infrastructure sous-jacente.

> **Note**
>
> Les blocs-notes Neptune, gérés via Amazon SageMaker AI, ne sont actuellement pas disponibles dans la région Asie-Pacifique (Malaisie) (ap-southeast-5). Cependant, vous pouvez toujours déployer des blocs-notes Neptune via d'autres options non gérées. Reportez-vous à Configuration manuelle des blocs-notes Neptune pour le déploiement manuel de blocs-notes.

Neptune fournit Jupyter et des JupyterLabblocs-notes dans le cadre du projet open source Neptune Graph Notebook sur et dans le plan de travail Neptune. GitHub Ces blocs-notes offrent des exemples de didacticiels d'application et des extraits de code dans un environnement de codage interactif où vous pouvez vous familiariser avec la technologie des graphes et Neptune. Vous pouvez les utiliser pour comprendre comment configurer, renseigner et interroger des graphes à l'aide de différents langages de requête, de différents jeux de données et même de différentes bases de données sur le backend.

Vous pouvez héberger ces blocs-notes de différentes manières :

- Le pupitre Neptune vous permet d'exécuter des blocs-notes Jupyter dans un environnement entièrement géré, hébergé dans Amazon SageMaker AI, et charge automatiquement la dernière version du projet de bloc-notes Neptune Graph pour vous. Il est facile de configurer le workbench dans la console Neptune lorsque vous créez une base de données Neptune.

  > **Note**
  >
  > Lorsque vous créez une instance de bloc-notes Neptune, deux options vous sont proposées pour accéder au réseau : un accès direct via Amazon SageMaker AI (par défaut) et un accès via un VPC. Quelle que soit l'option, le bloc-notes doit avoir accès à

Internet pour récupérer les dépendances des packages afin d'installer le pupitre Neptune. Le manque d'accès à Internet entraînera l'échec de la création d'une instance de bloc-notes Neptune.

- Vous pouvez également installer Jupyter localement. Cela vous permet d'exécuter les blocs-notes à partir de votre ordinateur portable connecté soit à Neptune, soit à une instance locale de l'une des bases de données orientées graphe open source. Dans ce dernier cas, vous pouvez expérimenter la technologie des graphes autant que vous le souhaitez sans avoir à dépenser un centime. Puis, lorsque vous serez prêt, vous pourrez passer en douceur à l'environnement de production géré proposé par Neptune.

Utilisation du workbench Neptune pour héberger des blocs-notes Neptune

Neptune propose des types d'instances T3 et T4g que vous pouvez utiliser pour moins de 0,10 USD de l'heure. Les ressources de workbench vous sont facturées SageMaker via Amazon AI, séparément de votre facturation Neptune. Consultez la page de tarification de Neptune. Jupyter et les JupyterLab blocs-notes créés sur le plan de travail Neptune utilisent tous un environnement Amazon Linux 2 et 4. JupyterLab Pour plus d'informations sur la prise en charge des JupyterLab ordinateurs portables, consultez la documentation Amazon SageMaker AI.

Vous pouvez créer un Jupyter ou un JupyterLab bloc-notes à l'aide du plan de travail Neptune de deux manières AWS Management Console :

- Utilisez le menu de configuration du bloc-notes lors de la création d'un cluster de bases de données Neptune. Pour cela, suivez les étapes décrites dans Lancement d'un cluster de base de données Neptune à l'aide du AWS Management Console.
- Utilisez le menu Blocs-notes dans le volet de navigation de gauche une fois que le cluster de bases de données a été créé. Pour cela, suivez les étapes ci-dessous.

Pour créer un Jupyter ou un JupyterLab bloc-notes à l'aide du menu Carnets

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le panneau de navigation de gauche, choisissez Bloc-notes.
3. Choisissez Create Notebook (Créer un bloc-notes).
4. Choisissez Database comme service Neptune.

5. Dans la liste Cluster, choisissez votre cluster de bases de données Neptune. Si vous n'avez pas encore de cluster de bases de données, choisissez Créer un cluster pour en créer un.
6. Sélectionnez un type d'instance de bloc-notes.
7. Donnez un nom à votre bloc-notes et éventuellement une description.
8. À moins que vous n'ayez déjà créé un rôle Gestion des identités et des accès AWS (IAM) pour vos blocs-notes, choisissez Créer un rôle IAM et entrez un nom de rôle IAM.

> **Note**
>
> Si vous choisissez de réutiliser un rôle IAM créé pour un bloc-notes précédent, la politique de rôle doit contenir les autorisations appropriées pour accéder au cluster de base de données Neptune que vous utilisez. Pour ce faire, vérifiez que les composants de l'ARN de la ressource sous l'action `neptune-db:*` correspondent à ce cluster. Des autorisations mal configurées entraînent des erreurs de connexion lorsque vous essayez d'exécuter les commandes de magie de bloc-notes.

9. Choisissez Create Notebook (Créer un bloc-notes). Le processus de création peut prendre de 5 à 10 minutes avant que tout soit prêt.
10. Une fois votre bloc-notes créé, sélectionnez-le, puis choisissez Open Jupyter ou Open. JupyterLab

La console peut créer un rôle Gestion des identités et des accès AWS (IAM) pour vos blocs-notes, ou vous pouvez en créer un vous-même. La politique pour ce rôle devrait comprendre les éléments suivants :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowS3GetObjectS3ListBucket",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
```

```
      ],
      "Resource": [
        "arn:aws:s3:::aws-neptune-notebook",
        "arn:aws:s3:::aws-neptune-notebook/*",
        "arn:aws:s3:::aws-neptune-notebook-us-east-1",
        "arn:aws:s3:::aws-neptune-notebook-us-east-1/*"
      ]
    },
    {
      "Sid": "AllowAccessNeptuneDB",
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": [
        "arn:aws:neptune-db:us-east-1:111122223333:cluster-resource-id/*"
      ]
    }
  ]
}
```

Notez que la deuxième déclaration de la politique ci-dessus répertorie une ou plusieurs ressources du cluster Neptune. IDs

En outre, le rôle doit établir la relation de confiance suivante :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Là aussi, la temps de préparation peut aller de 5 à 10 minutes.

Vous pouvez configurer le nouveau bloc-notes pour qu'il fonctionne avec Neptune ML, comme expliqué dans Configuration manuelle d'un bloc-notes Neptune pour Neptune ML.

Utiliser Python pour connecter un bloc-notes SageMaker IA générique à Neptune

Connecter un bloc-notes à Neptune est facile si vous avez installé Neptune magics, mais il est également possible de connecter un SageMaker bloc-notes intelligent à Neptune en utilisant Python, même si vous n'utilisez pas de bloc-notes Neptune.

Étapes à suivre pour se connecter à Neptune dans un bloc-notes doté d'une SageMaker intelligence artificielle

1. Installez le client Python Gremlin :

   ```
   !pip install gremlinpython
   ```

   Les blocs-notes Neptune installent le client Python Gremlin pour vous. Cette étape n'est donc nécessaire que si vous utilisez un bloc-notes doté d'une intelligence artificielle ordinaire. SageMaker

2. Écrivez un code tel que le suivant pour vous connecter et émettre une requête Gremlin :

   ```
   from gremlin_python import statics
   from gremlin_python.structure.graph import Graph
   from gremlin_python.process.graph_traversal import __
   from gremlin_python.process.strategies import *
   from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection
   from gremlin_python.driver.aiohttp.transport import AiohttpTransport
   from gremlin_python.process.traversal import *
   import os

   port = 8182
   server = '(your server endpoint)'

   endpoint = f'wss://{server}:{port}/gremlin'

   graph=Graph()

   connection = DriverRemoteConnection(endpoint,'g',
   ```

```
 transport_factory=lambda:AiohttpTransport(call_from_event_loop=True))

g = graph.traversal().withRemote(connection)

results = (g.V().hasLabel('airport')
                .sample(10)
                .order()
                .by('code')
                .local(__.values('code','city').fold())
                .toList())

# Print the results in a tabular form with a row index
for i,c in enumerate(results,1):
    print("%3d %4s %s" % (i,c[0],c[1]))

connection.close()
```

Note

S'il se trouve que vous utilisez une version du client Python Gremlin antérieure à la version 3.5.0, la ligne suivante :

```
connection = DriverRemoteConnection(endpoint,'g',

 transport_factory=lambda:AiohttpTransport(call_from_event_loop=True))
```

Serait simplement :

```
connection = DriverRemoteConnection(endpoint,'g')
```

Lancement de graph-notebook en tant qu'Amazon Neptune Workbench via CloudFormation

Vous pouvez déployer des blocs-notes Amazon Neptune Workbench à l'aide de. CloudFormation Le CloudFormation modèle déploie les blocs-notes Neptune Workbench en tant que ressources et inclut les carnets de base « Getting Started ». Le workbench vous permet de travailler avec votre cluster de base de données Amazon Neptune à l'aide de blocs-notes Jupyter hébergés par Amazon.

SageMaker Les ressources de Workbench vous sont facturées via Amazon SageMaker, séparément de votre facturation Neptune.

Le CloudFormation modèle et les instructions détaillées, y compris les exigences relatives aux rôles IAM et les autorisations, sont disponibles dans le référentiel graph-notebook GitHub .

Activation CloudWatch des journaux sur les ordinateurs portables Neptune

CloudWatch les journaux sont désormais activés par défaut pour les blocs-notes Neptune. Si vous possédez un ancien bloc-notes qui ne produit pas de CloudWatch journaux, procédez comme suit pour les activer manuellement :

1. Connectez-vous à la console SageMaker AI AWS Management Console et ouvrez-la.
2. Dans le panneau de navigation de gauche, choisissez Bloc-notes, puis Instances de blocs-notes. Recherchez le nom du bloc-notes Neptune pour lequel vous souhaitez activer les journaux.
3. Accédez à la page de détails en sélectionnant le nom de cette instance de bloc-notes.
4. Si l'instance du bloc-notes est en cours d'exécution, sélectionnez le bouton Arrêter en haut à droite de la page de détails du bloc-notes.
5. Sous Autorisations et chiffrement, il existe un champ pour l'ARN du rôle IAM. Sélectionnez le lien dans ce champ pour accéder au rôle IAM avec lequel cette instance de bloc-notes sera exécutée.
6. Créez la politique suivante :

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DeleteLogDelivery",
        "logs:Describe*",
        "logs:GetLogDelivery",
        "logs:GetLogEvents",
        "logs:ListLogDeliveries",
```

```
        "logs:PutLogEvents",
        "logs:PutResourcePolicy",
        "logs:UpdateLogDelivery"
      ],
      "Resource": "*"
    }
  ]
}
```

7. Enregistrez cette nouvelle politique et associez-la au rôle IAM indiqué à l'étape 4.
8. Cliquez sur Démarrer en haut à droite de la page de détails de l'instance SageMaker AI Notebook.
9. Lorsque les journaux commencent à circuler, vous devriez voir un lien Afficher les journaux sous le champ intitulé Configuration du cycle de vie en bas à gauche de la section Paramètres d'instance du bloc-notes de la page de détails.

Si un bloc-notes ne démarre pas, un message s'affichera sur la page de détails du bloc-notes de la console SageMaker AI, indiquant que l'instance du bloc-notes a mis plus de 5 minutes à démarrer. CloudWatch les journaux relatifs à ce problème se trouvent sous le nom suivant :

```
(your-notebook-name)/LifecycleConfigOnStart
```

Configuration de bloc-notes de graphe sur votre ordinateur local

Le projet graph-notebook contient des instructions pour configurer les blocs-notes Neptune sur votre ordinateur local :

- Conditions préalables
- Jupyter et installation JupyterLab
- Connexion à une base de données orientée graphe.

Vous pouvez connecter vos blocs-notes locaux soit à un cluster de bases de données Neptune, soit à une instance locale ou distante d'une base de données orientée graphe open source.

Utilisation des blocs-notes Neptune avec des clusters Neptune

Si vous vous connectez à un cluster Neptune sur le back-end, vous souhaiterez peut-être exécuter les blocs-notes dans Amazon AI. SageMaker La connexion à Neptune depuis l' SageMaker IA peut

être plus pratique que depuis une installation locale des ordinateurs portables, et cela vous permettra de travailler plus facilement avec Neptune ML.

Pour obtenir des instructions sur la configuration des blocs-notes dans SageMaker AI, consultez Launching graph-notebook using Amazon. SageMaker

Pour obtenir des instructions sur l'installation et la configuration de Neptune lui-même, consultez Configuration d'Amazon Neptune.

Vous pouvez également connecter une installation locale des blocs-notes Neptune à un cluster de bases de données Neptune. Cela peut être un peu plus compliqué, car les clusters de bases de données Amazon Neptune ne peuvent être créés que dans un réseau Amazon Virtual Private Cloud (VPC), qui est par nature isolé du monde extérieur. Il existe plusieurs manières de se connecter à un VPC depuis l'extérieur. L'une d'elles consiste à utiliser un équilibreur de charge. Une autre solution consiste à utiliser l'appairage de VPC (consultez le Guide de l'appairage Amazon Virtual Private Cloud).

Le moyen le plus pratique pour la plupart des utilisateurs est toutefois de se connecter pour configurer un serveur proxy Amazon EC2 au sein du VPC, puis d'utiliser le tunneling SSH (également appelé transfert de port) pour s'y connecter. Vous trouverez des instructions sur la procédure de configuration dans Connecting graph notebook locally to Amazon Neptune dans le `additional-databases/neptune` dossier du projet GitHub graph-notebook.

Utilisation des blocs-notes Neptune avec des bases de données orientées graphe open source

Pour commencer à utiliser gratuitement la technologie des graphes, vous pouvez également utiliser les blocs-notes Neptune avec diverses bases de données open source sur le backend. Le serveur TinkerPop Gremlin et la base de données Blazegraph en sont des exemples.

Pour utiliser le serveur Gremlin comme base de données principale, procédez comme suit :

- Le bloc-notes de connexion à un dossier du serveur Gremlin. GitHub
- Le dossier de configuration Gremlin du graph-notebook. GitHub

Pour utiliser une instance locale de Blazegraph comme base de données principale, procédez comme suit :

- Consultez les instructions de démarrage rapide de Blazegraph pour comprendre l'installation de base et la configuration requises pour exécuter une instance de Blazegraph.

- Accédez au GitHub dossier de configuration Graph-notebook Blazegraph contenant les fichiers et les instructions nécessaires pour configurer une instance Blazegraph locale.
- Dans le GitHub référentiel, accédez au répertoire « blazegraph » et suivez les instructions fournies pour configurer votre instance Blazegraph locale. Cela inclut les étapes de téléchargement du logiciel Blazegraph, de configuration des fichiers nécessaires et de démarrage du serveur Blazegraph.

Une fois qu'une instance Blazegraph locale est en cours d'exécution, vous pouvez l'intégrer à votre application en tant que base de données principale pour vos données et requêtes basées sur des graphes. Reportez-vous à la documentation et à l'exemple de code fournis dans le référentiel graph-notebook pour savoir comment connecter votre application à l'instance Blazegraph.

Migration des blocs-notes Neptune vers la version 4.x JupyterLab

Cette section décrit différentes approches pour migrer vos blocs-notes Neptune vers les environnements Amazon Linux 4.x et JupyterLab les versions plus récentes. Pour obtenir des informations détaillées sur le JupyterLab versionnement, consultez Amazon SageMaker AI JupyterLab Versioning.

Approches de migration

Nouvelle installation

Si vous n'avez pas besoin de conserver les fichiers ou les configurations d'espace de travail existants, vous pouvez :

1. Création d'une nouvelle instance de bloc-notes exécutant JupyterLab 4.x (notebook-al2-v3)
2. Vérifiez que la nouvelle configuration fonctionne comme prévu
3. Arrêtez et supprimez votre ancienne instance de bloc-notes

Migration par transfert de fichiers

Cette méthode utilise votre système local ou Amazon S3 comme stockage intermédiaire.

Idéal pour

- Accès direct à Internet via la configuration réseau Amazon SageMaker AI.

- Volume modéré de données à migrer
- Des fichiers spécifiques à préserver plutôt que des configurations d'espace de travail complètes.

Méthode 1 : utilisation de l' JupyterLab interface utilisateur

Idéal pour

- Petit nombre de fichiers
- Migration sélective de fichiers
- Préférez les drag-and-drop opérations simples

Étapes

1. Téléchargez des fichiers depuis l' JupyterLab instance source :
   - Naviguez et sélectionnez les fichiers que vous souhaitez migrer vers la nouvelle instance dans JupyterLab
   - Cliquez avec le bouton droit et sélectionnez Télécharger
2. Téléchargement vers une nouvelle JupyterLab instance :
   - Utilisez le bouton de téléchargement JupyterLab et sélectionnez tous les fichiers que vous souhaitez copier vers la nouvelle instance
   - (Ou) glissez et déposez des fichiers directement

Méthode 2 : utilisation d'Amazon S3

Idéal pour

- Grand nombre de fichiers
- Préservation de la structure de vos dossiers
- Migrations en masse

Conditions préalables

Assurez-vous que le rôle associé au bloc-notes dispose des autorisations appropriées pour télécharger et accéder au compartiment Amazon S3 :

```
{
"Effect": "Allow",
"Action": ["s3:PutObject", "s3:GetObject", "s3:ListBucket"],
"Resource": ["arn:aws:s3:::your-bucket-name/*", "arn:aws:s3:::your-bucket-name"]
}
```

> **Note**
>
> AWS La CLI doit être préinstallée sur les ordinateurs portables SageMaker AI.

Étapes

1. Ouvrez un terminal JupyterLab ou saisissez les commandes du terminal dans une cellule du bloc-notes avec un ! préfixe.
2. Copiez les fichiers de votre ancienne JupyterLab instance vers S3 à l'aide des commandes Amazon S3 cp ou Amazon S3 sync CLI :

   ```
   # using AWS s3 cp
   aws s3 cp /home/ec2-user/SageMaker/your-folder s3://your-bucket/backup/ --recursive

   # (OR) using AWS s3 sync
   aws s3 sync /home/ec2-user/SageMaker/your-folder s3://your-bucket/backup/
   ```

3. Copiez des fichiers depuis S3 vers votre nouvelle JupyterLab instance :

   ```
   # using AWS s3 cp
   aws s3 cp s3://your-bucket/backup/ /home/ec2-user/SageMaker/your-folder --recursive

   # (OR) using AWS s3 sync
   aws s3 sync s3://your-bucket/backup/ /home/ec2-user/SageMaker/your-folder
   ```

> **Note**
>
> `sync`À utiliser pour gérer les structures de dossiers, les mises à jour incrémentielles et `cp` pour les transferts ponctuels.

Migration vers Amazon EFS

Idéal pour

- Configuration réseau VPC uniquement
- Grands volumes de données

Étapes

Suivez le blog Mount an EFS file system to a Amazon SageMaker AI notebook pour utiliser un système de fichiers Amazon EFS avec vos instances de bloc-notes.

En outre, quelques étapes supplémentaires s'appliquent spécifiquement à la migration des blocs-notes Neptune vers le nouvel environnement :

1. Lors de la création du bloc-notes Neptune dans la console, sélectionnez Créer une nouvelle configuration du cycle de vie sous Configuration du cycle de vie
2. Dans la configuration du cycle de vie du modèle, ajoutez votre commande de montage Amazon EFS (`sudo mount -t nfs ...`) après le script install.sh

Cela garantit que votre système de fichiers Amazon EFS est automatiquement monté chaque fois que votre instance de bloc-notes démarre ou redémarre. Pour résoudre les problèmes de montage, consultez le document de résolution des problèmes d'Amazon EFS.

Avantages

- Accès fluide aux fichiers dans toutes les instances
- Accès direct aux fichiers sans transferts intermédiaires
- Gestion efficace de grands ensembles de données

Migration de volumes Amazon EBS

Idéal lorsque vous devez conserver

- Configurations complètes de l'espace

- Fichiers cachés
- Réglages du système
- Préservation des configurations complètes de l'espace de travail, des fichiers cachés et des paramètres système

Suivez le guide de migration basé sur l'AWS SageMaker IA pour les volumes Amazon EBS afin de transférer des fichiers depuis le volume Amazon EBS associé aux instances de bloc-notes.

En outre, quelques étapes supplémentaires s'appliquent spécifiquement à la migration des blocs-notes Neptune vers le nouvel environnement.

Prérequis spécifiques à Neptune

Dans le rôle IAM du bloc-notes Neptune source, ajoutez toutes les autorisations suivantes :

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:ListBucket",
    "s3:CreateBucket",
    "s3:PutObject"
  ],
  "Resource": [
    "arn:aws:s3:::(your ebs backup bucket name)",
    "arn:aws:s3:::(your ebs backup bucket name)/*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "sagemaker:ListTags"
  ],
  "Resource": [
    "*"
  ]
}
```

Assurez-vous de spécifier l'ARN correct pour le compartiment S3 que vous utiliserez pour la sauvegarde.

Configuration du cycle de vie spécifique à Neptune

Lors de la création du deuxième script de configuration du cycle de vie pour restaurer la sauvegarde (à partir de `on-create.sh`), comme décrit dans le billet de blog, le nom du cycle de vie doit suivre le format `aws-neptune-*`, comme `aws-neptune-sync-from-s3`. Cela garantit que le LCC peut être sélectionné lors de la création du bloc-notes dans la console Neptune.

Synchronisation spécifique à Neptune entre un instantané et une nouvelle instance

Dans les étapes décrites dans le billet de blog pour la synchronisation d'un instantané vers une nouvelle instance, voici les modifications spécifiques à Neptune :

- À l'étape 4, choisissez notebook-al2-v3.
- À l'étape 5, réutilisez le rôle IAM du bloc-notes Neptune source.
- Entre les étapes 7 et 8 :
  - Dans Paramètres d'instance du bloc-notes, définissez un nom utilisant le format `aws-neptune-*`.
  - Ouvrez l'accordéon des paramètres réseau et sélectionnez le même VPC, le même sous-réseau et le même groupe de sécurité que dans le bloc-notes source.

Étapes spécifiques à Neptune après la création du bloc-notes

1. Sélectionnez le bouton Ouvrir Jupyter correspondant au bloc-notes. Une fois que le fichier `SYNC_COMPLETE` apparaît dans le répertoire principal, passez à l'étape suivante.
2. Accédez à la page de l'instance du bloc-notes dans la console SageMaker AI.
3. Arrêtez le bloc-notes.
4. Tâche de sélection Modifier.
5. Dans les paramètres d'instance du bloc-notes, modifiez le champ Configuration du cycle de vie en sélectionnant le cycle de vie original du bloc-notes Neptune source. Notez qu'il ne s'agit pas du cycle de vie de sauvegarde EBS.
6. Sélectionnez Mettre à jour les paramètres du bloc-notes.
7. Redémarrez le bloc-notes.

Avec les modifications décrites ici par rapport aux étapes décrites dans le billet de blog, vos blocs-notes graphiques devraient désormais être migrés vers une nouvelle instance de bloc-notes Neptune qui utilise l'environnement Amazon Linux 2 et 4. JupyterLab Ils seront accessibles et gérés sur la page Neptune du AWS Management Console, et vous pouvez maintenant reprendre votre travail là où vous vous êtes arrêté en sélectionnant Open Jupyter ou Open. JupyterLab

Création d'un bloc-notes Neptune dans les instances Amazon AI SageMaker

1. Ouvrez la console Amazon SageMaker AI à l'adresse https://console.aws.amazon.com/sagemaker/.
2. Dans le volet de navigation, développez Notebook, puis choisissez Notebook instances.
3. Choisissez Create notebook instance (Créer une instance de bloc-notes).
4. Dans les paramètres de l'instance du bloc-notes, sous Nom de l'instance du bloc-notes, attribuez au bloc-notes un nom préfixé par `aws-neptune-` (par exemple,`aws-neptune-my-test-notebook`).
5. Sous Identifiant de plate-forme, sélectionnez Amazon Linux 2, JupyterLab 4.
6. Sélectionnez Configuration supplémentaire.
7. Sous Configuration du cycle de vie, choisissez Créer une nouvelle configuration du cycle de vie.
8. Dans Configuration, sous Nom, entrez le nom de l'instance du bloc-notes indiqué à l'étape 4.
9. Dans Scripts, sous Carnet de démarrage, remplacez le script existant par celui-ci :

```
#!/bin/bash

sudo -u ec2-user -i <<'EOF'

echo "export GRAPH_NOTEBOOK_AUTH_MODE=IAM" >> ~/.bashrc
echo "export GRAPH_NOTEBOOK_SSL=True" >> ~/.bashrc
echo "export GRAPH_NOTEBOOK_SERVICE=neptune-db for Neptune, or neptune-graph for
 Neptune Analytics" >> ~/.bashrc
echo "export GRAPH_NOTEBOOK_HOST=(Neptune Analytics graph endpoint, public or
 private)" >> ~/.bashrc
echo "export GRAPH_NOTEBOOK_PORT=8182" >> ~/.bashrc
echo "export NEPTUNE_LOAD_FROM_S3_ROLE_ARN=" >> ~/.bashrc
echo "export AWS_REGION=(AWS region)" >> ~/.bashrc
```

```
aws s3 cp s3://aws-neptune-notebook-(AWS region)/graph_notebook.tar.gz /tmp/
graph_notebook.tar.gz
rm -rf /tmp/graph_notebook
tar -zxvf /tmp/graph_notebook.tar.gz -C /tmp
/tmp/graph_notebook/install_jl4x.sh

EOF
```

10. Sélectionnez Create configuration (Créer une configuration).
11. Dans Autorisations et chiffrement, sous Rôle IAM, sélectionnez le rôle que vous avez créé ci-dessus.
12. En réseau, si vous utilisez un point de terminaison de graphe privé :
    a. Sous VPC, sélectionnez le VPC dans lequel réside le graphe Neptune Analytics.
    b. Sous Sous-réseau, sélectionnez un sous-réseau associé au graphe Neptune Analytics.
    c. Sous Groupe (s) de sécurité, sélectionnez tous les groupes de sécurité associés au graphe Neptune Analytics.
13. Choisissez Create notebook instance (Créer une instance de bloc-notes).
14. Après 5 ou 10 minutes, lorsque votre nouveau bloc-notes atteint `Ready` son statut, sélectionnez-le. Choisissez Open Jupyter ou Open. JupyterLab

Configuration manuelle des blocs-notes Neptune

Vous pouvez également utiliser les packages AWS open source disponibles pour graph-notebook et graph-explorer pour configurer un environnement de bloc-notes Neptune. Bien qu'il existe plusieurs manières de configurer un bloc-notes à l'aide de packages open source, l'approche recommandée consiste à :

- Configurez un bloc-notes sur votre machine locale et un tunnel SSH Amazon EC2 qui connecte votre machine locale à un cluster Neptune sur le même VPC que l'instance Amazon EC2.
- Configurez Graph-Explorer sur une instance Amazon EC2 au sein de votre VPC.

Pour obtenir des instructions détaillées sur la configuration des packages open-source graph-notebooks et graph-explorer, reportez-vous à la documentation officielle et aux référentiels suivants : AWS GitHub

- https://docs.aws.amazon.com//neptune/latest/userguide/graph-notebooks.html#graph-notebooks-local
- https://docs.aws.amazon.com//neptune/latest/userguide/get-started-connect-ec2-same-vpc.html
- https://github.com/aws/graph-notebook/tree/main/additional-databases/neptune
- https://github.com/aws/graph-explorer/blob/main/additionaldocs/getting-started/README.md#amazon-ec2-setup

Configuration de sécurité

Lorsque vous configurez votre environnement, assurez-vous que les configurations de sécurité suivantes sont définies en conséquence :

- Groupe de sécurité du cluster Neptune : autorisez le trafic TCP entrant sur le port 8182 depuis le groupe de sécurité de votre instance Amazon EC2.
- Groupe de sécurité des instances Amazon EC2 : configurez les règles HTTPS entrantes pour permettre l'accès à Graph Explorer.

Nous vous recommandons d'utiliser un seul groupe de sécurité pour votre cluster Neptune et votre instance Amazon EC2. Cela simplifie la configuration et réduit les problèmes de connectivité potentiels. Cependant, pensez à ajouter une règle d'autoréférencement pour permettre la communication entre les ressources au sein du groupe.

Utilisation de la magie dans les blocs-notes Amazon Neptune

Le workbench Neptune fournit diverses commandes dites magiques dans les blocs-notes, qui permettent d'économiser beaucoup de temps et d'efforts. Elles se répartissent en deux catégories : la magie linéaire et la magie cellulaire.

La magie linéaire implique des commandes précédées d'un signe de pourcentage (%). Elles n'acceptent que des entrées de ligne, et non des entrées provenant du reste du corps des cellules. Le workbench Neptune fournit la magie linéaire suivante :

- %seed
- %load
- %load_ids
- %load_status

- %cancel_load
- %status
- %gremlin_status
- %opencypher_status ou %oc_status
- %stream_viewer
- %sparql_status
- %graph_notebook_config
- %graph_notebook_host
- %graph_notebook_version
- %graph_notebook_service
- %graph_notebook_vis_options
- %statistics
- %summary
- %reset_graph
- %get_graph
- % de distribution des diplômes

La magie cellulaire implique des commandes précédées de deux signes de pourcentahe (%%) au lieu d'un seul. Elle utilise le contenu de la cellule comme entrée, bien qu'elle accepte également le contenu de la ligne comme entrée. Le workbench Neptune fournit la magie cellulaire suivante :

- %%sparql
- %%gremlin
- %%opencypher ou %%oc
- %%graph_notebook_config
- %%graph_notebook_vis_options

Il existe également deux formes de magie, une magie linéaire et une magie cellulaire, pour travailler avec Machine learning Neptune :

- %neptune_ml

- %%neptune_ml

> **Note**
>
> Lorsque vous travaillez avec les commandes magiques Neptune, vous pouvez généralement obtenir un texte d'aide à l'aide d'un paramètre `--help` ou `-h`. Avec la magie cellulaire, le corps ne peut pas être vide. Ainsi, lorsque vous souhaitez obtenir de l'aide, insérez du texte de remplissage, ne serait-ce qu'un seul caractère, dans le corps. Par exemple :
>
> ```
> %%gremlin --help
>   x
> ```

Injection de variable dans la magie linéaire ou cellulaire

Les variables définies dans un bloc-notes peuvent être référencées dans n'importe quelle magie linéaire ou cellulaire du bloc-notes avec le format `${VAR_NAME}`.

Par exemple, supposons que vous définissiez les variables suivantes :

```
c = 'code'
my_edge_labels = '{"route":"dist"}'
```

Puis, cette requête Gremlin dans une magie cellulaire :

```
%%gremlin -de $my_edge_labels
g.V().has('${c}','SAF').out('route').values('${c}')
```

Équivaut à :

```
%%gremlin -de {"route":"dist"}
g.V().has('code','SAF').out('route').values('code')
```

Arguments de requête compatibles avec tous les langages de requête

Les arguments de requête suivants fonctionnent avec les commandes magiques `%%gremlin`, `%%opencypher` et `%%sparql` dans le workbench Neptune :

Arguments de requête courants

- **`--store-to`** (ou **`-s`**) : spécifie le nom d'une variable dans laquelle stocker les résultats de la requête.
- **`--silent`** : s'il est présent, aucun résultat n'est affiché une fois la requête terminée.

- **`--group-by`** (ou **`-g`**) : spécifie la propriété utilisée pour regrouper les nœuds (telle que `code` ou `T.region`). Les sommets sont colorés en fonction du groupe qui leur est attribué.
- **`--ignore-groups`** : s'il est présent, toutes les options de regroupement sont ignorées.
- **`--display-property`** (ou **`-d`**) : spécifie la propriété dont la valeur doit être affichée pour chaque sommet.

  La valeur par défaut pour chaque langage de requête est la suivante :

  - Pour Gremlin : `T.label`.
  - Pour openCypher : `~labels`.
  - Pour SPARQL : `type`.
- **`--edge-display-property`** (ou **`-t`**) : spécifie la propriété dont la valeur doit être affichée pour chaque arête.

  La valeur par défaut pour chaque langage de requête est la suivante :

  - Pour Gremlin : `T.label`.
  - Pour openCypher : `~labels`.
  - Pour SPARQL : `type`.
- **`--tooltip-property`** (ou **`-de`**) : spécifie une propriété dont la valeur doit être affichée sous forme d'infobulle pour chaque nœud.

  La valeur par défaut pour chaque langage de requête est la suivante :

  - Pour Gremlin : `T.label`.
  - Pour openCypher : `~labels`.
  - Pour SPARQL : `type`.
- **`--edge-tooltip-property`** (ou **`-te`**) : spécifie une propriété dont la valeur doit être affichée sous forme d'infobulle pour chaque arête.

  La valeur par défaut pour chaque langage de requête est la suivante :

  - Pour Gremlin : `T.label`.

  - Pour openCypher : `~labels`.
  - Pour SPARQL : `type`.
- **`--label-max-length`** (ou **`-l`**) : spécifie la longueur maximale de caractères d'une étiquette de sommet. La valeur par défaut est 10.
- **`--edge-label-max-length`** (ou **`-le`**) : spécifie la longueur maximale de caractères d'une étiquette d'arête. La valeur par défaut est 10.

  Dans le cas d'openCypher uniquement, `--rel-label-max-length` ou `-rel` est utilisé à la place.
- **`--simulation-duration`** (ou **`-sd`**) : spécifie la durée maximale de la simulation de la physique de visualisation. La valeur par défaut est 1 500 ms.
- **`--stop-physics`** (ou **`-sp`**) : désactive la physique de visualisation une fois la simulation initiale stabilisée.

Les valeurs de propriété de ces arguments peuvent consister en une clé de propriété unique ou en une chaîne JSON qui peut spécifier une propriété différente pour chaque type d'étiquette. Une chaîne JSON ne peut être spécifiée qu'à l'aide de l'injection de variables.

La magie linéaire **`%seed`**

La magie linéaire `%seed` est un moyen pratique d'ajouter des données au point de terminaison Neptune. Vous pouvez l'utiliser pour explorer et expérimenter des requêtes avec Gremlin, openCypher ou SPARQL. Elle fournit un formulaire dans lequel vous pouvez sélectionner le modèle de données que vous souhaitez explorer (graphe de propriétés ou RDF), puis choisir parmi un certain nombre d'exemples de jeux de données fournis par Neptune.

La magie linéaire **`%load`**

La magie linéaire `%load` génère un formulaire que vous pouvez utiliser pour soumettre une demande de chargement en bloc à Neptune (voir Commande de chargeur Neptune). La source doit être un chemin d'accès Amazon S3 dans la même région que le cluster Neptune.

La magie linéaire **`%load_ids`**

La magie linéaire `%load_ids` récupère les identifiants de chargement qui ont été soumis au point de terminaison hôte du bloc-notes (voir Paramètres de demande du Neptune Loader Get-Status). La demande prend la forme suivante :

```
GET https://your-neptune-endpoint:port/loader
```

La magie linéaire `%load_status`

La magie linéaire `%load_status` permet de récupérer le statut d'une tâche de chargement particulière qui a été soumise au point de terminaison hôte du bloc-notes, spécifié par la l'entrée de ligne (voir Paramètres de demande du Neptune Loader Get-Status). La demande prend la forme suivante :

```
GET https://your-neptune-endpoint:port/loader?loadId=loadId
```

La magie linéaire ressemble à ceci :

```
%load_status load id
```

La magie linéaire `%cancel_load`

La magie linéaire `%cancel_load` annule une tâche de chargement particulière (voir Tâche d'annulation du chargeur Neptune). La demande prend la forme suivante :

```
DELETE https://your-neptune-endpoint:port/loader?loadId=loadId
```

La magie linéaire ressemble à ceci :

```
%cancel_load load id
```

La magie linéaire `%status`

Récupère les informations de statut du point de terminaison hôte du bloc-notes (%graph_notebook_config affiche le point de terminaison hôte).

Pour les hôtes de base de données Neptune, les informations d'état seront extraites du point de terminaison de l'état de santé. Pour les hôtes de Neptune Analytics, le statut sera récupéré via l'GetGraph API. Pour plus d'informations, consultez %get_graph.

La magie linéaire `%gremlin_status`

Récupère les informations relatives au statut des requête Gremlin.

Magie linéaire **%opencypher_status** (**%oc_status** également)

Récupère le statut d'une requête openCypher. Cette magie linéaire accepte les arguments facultatifs suivants :

- **--queryId** ou **-q** : spécifie l'ID d'une requête spécifique en cours d'exécution dont le statut doit être affiché.
- **--cancelQuery** ou **-c** : annuler une requête en cours d'exécution. N'utilise aucune valeur.
- **--silent-cancel** ou **-s** : si `--silent` est défini sur `true` lors de l'annulation d'une requête, la requête en cours est annulée avec le code de réponse HTTP `200`. Dans le cas contraire, le code de réponse HTTP est `500`.
- **--store-to** : spécifie le nom d'une variable dans laquelle stocker les résultats de la requête.
- **-w/--includeWaiting**— Neptune DB uniquement. Lorsqu'il est défini sur true et que les autres paramètres ne sont pas présents, cela entraîne le renvoi des informations d'état pour les requêtes en attente ainsi que pour les requêtes en cours d'exécution. Ce paramètre n'est pas défini sur une valeur.
- **--state**— Neptune Analytics uniquement. Spécifie le sous-ensemble d'états de requête dont il faut récupérer le statut.
- **-m/--maxResults**— Neptune Analytics uniquement. Définit une limite supérieure pour l'ensemble de requêtes renvoyées correspondant à la valeur de`--state`.
- **--silent** : s'il est présent, aucun résultat n'est affiché une fois la requête terminée.

La magie linéaire **%stream_viewer**

La magie linéaire `%stream_viewer` affiche une interface qui permet d'explorer de manière interactive les entrées journalisées dans les flux Neptune, si les flux sont activés sur le cluster Neptune. Les arguments facultatifs suivants sont acceptés :

- **language** : langage de requête des données du flux (`gremlin` ou `sparql`). Si vous ne fournissez pas cet argument, la valeur par défaut est `gremlin`.
- **--limit** : spécifie le nombre maximal d'entrées de flux à afficher par page. Si vous ne fournissez pas cet argument, la valeur par défaut est `10`.

> **Note**
>
> La magie linéaire `%stream_viewer` n'est entièrement prise en charge que sur les versions 1.0.5.1 et antérieures du moteur.

La magie linéaire `%sparql_status`

Récupère les informations relatives au statut des requêtes SPARQL.

La magie linéaire `%graph_notebook_config`

Cette magie linéaire affiche un objet JSON contenant la configuration utilisée par le bloc-notes pour communiquer avec Neptune. La configuration inclut :

- `host` : point de terminaison auquel se connecter et vers lequel émettre des commandes.
- `port` : port utilisé lors de l'envoi de commandes à Neptune. La valeur par défaut est `8182`.
- `auth_mode` : mode d'authentification à utiliser lors de l'envoi de commandes à Neptune. Doit être `IAM` en cas de connexion à un cluster sur lequel l'authentification IAM est activée. `DEFAULT`, dans le cas contraire.
- `load_from_s3_arn` : spécifie un ARN Amazon S3 à utiliser par la magie `%load`. Si cette valeur est vide, l'ARN doit être spécifié dans la commande `%load`.
- `ssl` : valeur booléenne indiquant la connexion à Neptune à l'aide du protocole TLS ou non. La valeur par défaut est `true`.
- `aws_region` : région dans laquelle ce bloc-notes est déployé. Ces informations sont utilisées pour l'authentification IAM et pour les demandes `%load`.

Vous pouvez modifier la configuration en copiant la sortie `%graph_notebook_config` dans une nouvelle cellule et en y apportant des modifications. Ensuite, si vous exécutez la magie cellulaire %%graph_notebook_config au niveau de la nouvelle cellule, la configuration sera modifiée en conséquence.

La magie linéaire `%graph_notebook_host`

Définit l'entrée de ligne comme hôte du bloc-notes.

La magie linéaire `%graph_notebook_version`

La magie linéaire `%graph_notebook_version` renvoie le numéro de version du bloc-notes du workbench Neptune. Par exemple, la visualisation de graphes a été introduite dans la version `1.27`.

La magie linéaire `%graph_notebook_service`

La magie des `%graph_notebook_service` lignes définit l'entrée de ligne comme nom de service utilisé pour les requêtes Neptune.

La magie linéaire `%graph_notebook_vis_options`

La magie linéaire `%graph_notebook_vis_options` affiche les paramètres de visualisation actuels utilisés par le bloc-notes. Ces options sont expliquées dans la documentation du fichier vis.js.

Vous pouvez modifier ces paramètres en copiant la sortie dans une nouvelle cellule, en apportant les modifications souhaitées, puis en exécutant la magie cellulaire `%%graph_notebook_vis_options` au niveau de la cellule.

Pour restaurer les valeurs par défaut des paramètres de visualisation, vous pouvez exécuter la magie linéaire `%graph_notebook_vis_options` avec un paramètre `reset`. Tous les paramètres de visualisation sont ainsi réinitialisés :

```
%graph_notebook_vis_options reset
```

La magie linéaire `%statistics`

La magie linéaire `%statistics` est utilisée pour récupérer ou gérer les statistiques du moteur DFE (voir Gestion des statistiques à utiliser par le DFE Neptune). Cette magie peut également être utilisée pour récupérer un résumé de graphe.

Les paramètres suivants sont acceptés :

- **`--language`** : langage de requête du point de terminaison des statistiques : `propertygraph` (ou `pg`) ou `rdf`.

  Si ce paramètre n'est fourni, la valeur par défaut est `propertygraph`.
- **`--mode`** (ou **`-m`**) : spécifie le type de demande ou d'action à soumettre (`status`, `disableAutoCompute`, `enableAutoCompute`, `refresh`, `delete`, `detailed` ou `basic`).

Si cet argument n'est pas fourni, la valeur par défaut est `status`, sauf si `--summary` est spécifie, auquel cas la valeur par défaut est `basic`.

- **`--summary`** : récupère le résumé du graphe depuis le point de terminaison de résumé des statistiques correspondant au langage sélectionné.
- **`--silent`** : s'il est présent, aucun résultat n'est affiché une fois la requête terminée.
- **`--store-to`** : utilisé pour spécifier une variable dans laquelle stocker les résultats de la requête.

La magie linéaire **`%summary`**

La magie linéaire `%summary` est utilisée pour récupérer les informations de résumé de graphe. Elle est disponible à partir de la version `1.2.1.0` du moteur Neptune.

Les paramètres suivants sont acceptés :

- **`--language`** : langage de requête du point de terminaison des statistiques : `propertygraph` (ou `pg`) ou `rdf`.

  Si ce paramètre n'est fourni, la valeur par défaut est `propertygraph`.
- **`--detailed`** : active ou désactive l'affichage des champs de structures dans la sortie.

  Si cet argument n'est pas fourni, le mode d'affichage de résumé `basic` est le mode par défaut.
- **`--silent`** : s'il est présent, aucun résultat n'est affiché une fois la requête terminée.
- **`--store-to`** : utilisé pour spécifier une variable dans laquelle stocker les résultats de la requête.

La magie linéaire **`%reset_graph`**

La magie de la ligne `%reset_graph` (ou`%_graph_reset`) exécute un ResetGraphappel sur le point de terminaison Neptune Analytics. Elle accepte l'entrée de ligne facultative suivante :

- -ns ou -- no-skip-snapshot - Le cas échéant, un instantané final du graphe sera créé avant que les données du graphe ne soient supprimées.
- --silent — S'il est présent, aucune sortie n'est affichée après l'envoi de l'appel de réinitialisation.
- --store-to — Utilisé pour spécifier une variable dans laquelle stocker la réponse. ResetGraph

La magie linéaire `%get_graph`

La magie des `%get_graph` lignes récupère des informations sur un graphe via l'GetGraph API. Cette magie est fonctionnellement identique à celle %status utilisée avec Neptune Analytics.

La magie linéaire `%degreeDistribution`

La magie des `%degreeDistribution` lignes crée un histogramme interactif qui permet de visualiser la distribution des degrés des sommets dans un graphique. L'histogramme affiche le nombre de sommets (axe Y) ayant un degré spécifique (axe X). Il n'est disponible qu'avec Neptune Analytics.

Il accepte les paramètres suivants, qui peuvent également être choisis dans le menu déroulant :

- **`--traversalDirection`**— Spécifie le degré à analyser. Doit être l'une des valeurs suivantes : `"both"` (par défaut)`"inbound"`, ou`"outbound"`.
- **`--vertexLabels`**— Filtrez les sommets en fonction des étiquettes fournies (par exemple,`airport country`). La valeur par défaut est une liste vide.
- **`--edgeLabels`**— Filtrez les bords en fonction des étiquettes fournies (par exemple,`route`). La valeur par défaut est une liste vide.

La magie cellulaire `%%sparql`

La magie cellulaire `%%sparql` envoie une requête SPARQL au point de terminaison Neptune. Elle accepte l'entrée de ligne facultative suivante :

- **`-h`** ou **`--help`** : renvoie un texte d'aide concernant ces paramètres.
- **`--path`** : ajoute un préfixe à un chemin d'accès au point de terminaison SPARQL. Par exemple, si vous spécifiez `--path "abc/def"`, le point de terminaison appelé est *`host`*`:`*`port`*`/abc/def`.
- **`--expand-all`** : indicateur de visualisation des requêtes qui indique au visualiseur d'inclure tous les résultats `?s ?p ?o` dans le schéma du graphe, quel que soit le type de liaison.

  Par défaut, une visualisation SPARQL inclut uniquement des modèles triples où `o?` est un `uri` ou un nœud vide (`bnode`). Tous les autres types de liaison `?o`, tels que les chaînes littérales ou les entiers, sont traités comme des propriétés du nœud `?s` qui peuvent être visualisées dans le volet Details (Détails) de l'onglet Graph (Graphe).

Utilisez l'indicateur de requête `--expand-all` lorsque vous souhaitez plutôt inclure des valeurs littérales telles que des sommets dans la visualisation.

Ne combinez pas cet indicateur de visualisation avec les paramètres explain, car les requêtes explain ne sont pas visualisées.

- **`--explain-type`** : utilisé pour spécifier le mode explain à utiliser (`dynamic`, `static` ou `details`).
- **`--explain-format`** : utilisé pour spécifier le format de réponse pour une requête explain (`text/csv` ou `text/html`).
- **`--store-to`** : utilisé pour spécifier une variable dans laquelle stocker les résultats de la requête.

Exemple de requête `explain` :

```
%%sparql explain

SELECT * WHERE {?s ?p ?o} LIMIT 10
```

Exemple de requête de visualisation avec un indicateur de requête de visualisation `--expand-all` (voir Visualisation SPARQL) :

```
%%sparql --expand-all

SELECT * WHERE {?s ?p ?o} LIMIT 10
```

La magie cellulaire **`%%gremlin`**

La magie `%%gremlin` cellulaire envoie une requête G705 au point de terminaison Neptune à l'aide de. WebSocket Elle accepte une entrée de ligne facultative pour passer en mode Gremlin `explain` / > ou API Gremlin `profile`, ainsi qu'une entrée d'indicateur de visualisation facultative séparée pour modifier le comportement de sortie de visualisation (voir Visualisation Gremlin).

Exemple de requête `explain` :

```
%%gremlin explain

g.V().limit(10)
```

Exemple de requête `profile` :

```
%%gremlin profile

g.V().limit(10)
```

Exemple de requête de visualisation avec un indicateur de requête de visualisation :

```
%%gremlin -p v,outv

g.V().out().limit(10)
```

Paramètres facultatifs pour les requêtes **`%%gremlin profile`**

- **`--profile-chop`** : spécifie la longueur maximale de la chaîne de résultats de la fonctionnalité profile. La valeur par défaut si vous ne fournissez pas cet argument est 250.
- **`--profile-serializer`** : spécifie le sérialiseur à utiliser pour les résultats. Les valeurs autorisées sont toutes les valeurs d'énumération valides du type MIME ou TinkerPop du pilote « Serializers ». La valeur par défaut si vous ne fournissez pas cet argument est `application.json`.
- **`--profile-no-results`** : affiche uniquement le nombre de résultats. Si cet argument n'est pas utilisé, tous les résultats de la requête sont affichés par défaut dans le rapport de la fonctionnalité profile.
- **`--profile-indexOps`** : affiche un rapport détaillé de toutes les opérations d'indexation.

Magie cellulaire **`%%opencypher`** (**`%%oc`** également)

La magie cellulaire `%%opencypher` (qui a également une forme `%%oc` abrégée) envoie une requête openCypher au point de terminaison Neptune. Elle accepte les arguments d'entrée de ligne facultatifs suivants :

- **`--mode`**— Le mode de requête : `query` soit `bolt` La valeur par défaut si vous ne fournissez pas cet argument est `query`.
- **`--group-by`** ou **`-g`** : spécifie la propriété utilisée pour regrouper les nœuds. Par exemple, `code, ~id`. La valeur par défaut si vous ne fournissez pas cet argument est `~labels`.
- **`--ignore-groups`** : s'il est présent, toutes les options de regroupement sont ignorées.

- **`--display-property`** ou **`-d`** : spécifie la propriété dont la valeur doit être affichée pour chaque sommet. La valeur par défaut si vous ne fournissez pas cet argument est `~labels`.
- **`--edge-display-property`** ou **`-de`** : spécifie la propriété dont la valeur doit être affichée pour chaque arête. La valeur par défaut si vous ne fournissez pas cet argument est `~labels`.
- **`--explain-type`**— Utilisé pour spécifier le mode d'explication à utiliser (dynamique, statique, debug ou details).
- **`--label-max-length`** ou **`-l`** : spécifie le nombre maximal de caractères d'une étiquette de sommet à afficher. La valeur par défaut si vous ne fournissez pas cet argument est `10`.
- **`--store-to`** ou **`-s`** : spécifie le nom d'une variable dans laquelle stocker les résultats de la requête.
- **`--plan-cache`** ou **`-pc`** : spécifie le mode de cache du plan à utiliser. La valeur par défaut est `auto`.
- **`--query-timeout`** ou **`-qt`** : spécifie le délai d'expiration maximal des requêtes en millisecondes. La valeur par défaut est `1800000`.
- **`--query-parameters`** ou **`qp`** : Définitions de paramètres à appliquer à la requête. Cette option peut accepter un nom de variable unique ou une représentation sous forme de chaîne de la carte.

  Exemple d'utilisation de **`--query-parameters`**

  1. Définissez une carte des paramètres openCypher dans une cellule du bloc-notes.

     ```
     params = '''{
      "name":"john",
      "age": 20,
     }'''
     ```

  2. Transmettez les paramètres `--query-parameters` dans une autre cellule avec `%%oc`.

     ```
     %%oc --query-parameters params

     MATCH (n {name: $name, age: $age})
     RETURN n
     ```

La magie cellulaire **`%%graph_notebook_config`**

La magie cellulaire `%%graph_notebook_config` utilise un objet JSON contenant des informations de configuration pour modifier les paramètres utilisés par le bloc-notes pour communiquer avec

Neptune, si possible. La configuration prend la même forme que celle renvoyée par la magie linéaire %graph_notebook_config.

Par exemple :

```
%%graph_notebook_config
{
  "host": "my-new-cluster-endpoint.amazon.com",
  "port": 8182,
  "auth_mode": "DEFAULT",
  "load_from_s3_arn": "",
  "ssl": true,
  "aws_region": "us-east-1"
}
```

La magie cellulaire `%%graph_notebook_vis_options`

La magie cellulaire `%%graph_notebook_vis_options` vous permet de définir les options de visualisation du bloc-notes. Vous pouvez copier les paramètres renvoyés par la magie linéaire `%graph-notebook-vis-options` dans une nouvelle cellule, les modifier et utiliser la magie cellulaire `%%graph_notebook_vis_options` pour définir de nouvelles valeurs.

Ces options sont expliquées dans la documentation du fichier vis.js.

Pour restaurer les valeurs par défaut des paramètres de visualisation, vous pouvez exécuter la magie linéaire `%graph_notebook_vis_options` avec un paramètre `reset`. Tous les paramètres de visualisation sont ainsi réinitialisés :

```
%graph_notebook_vis_options reset
```

La magie linéaire `%neptune_ml`

Vous pouvez utiliser la magie linéaire `%neptune_ml` pour lancer et gérer diverses opérations Neptune ML.

> **Note**
>
> Vous pouvez également lancer et gérer certaines opérations Neptune ML à l'aide de la magie cellulaire %%neptune_ml.

- **`%neptune_ml export start`** : démarre une nouvelle tâche d'exportation.

  Parameters

  - **`--export-url`***`exporter-endpoint`*— (facultatif) Le point de terminaison Amazon API Gateway où l'exportateur peut être appelé.
  - **`--export-iam`** : (facultatif) indicateur signalant que les demandes adressées à l'URL d'exportation doivent être signées à l'aide de SigV4.
  - **`--export-no-ssl`** : (facultatif) indicateur signalant que le protocole SSL ne doit pas être utilisé lors de la connexion à l'exportateur.
  - **`--wait`** : (facultatif) indicateur signalant que l'opération doit attendre la fin de l'exportation.
  - **`--wait-interval`***`interval-to-wait`*— (facultatif) Définit le délai, en secondes, entre les vérifications du statut d'exportation (par défaut : 60).
  - **`--wait-timeout`***`timeout-seconds`*— (facultatif) Définit le temps, en secondes, à attendre que la tâche d'exportation soit terminée avant de renvoyer le statut le plus récent (par défaut : 3 600).
  - **`--store-to`***`location-to-store-result`*— (facultatif) Variable dans laquelle stocker le résultat de l'exportation. Si `--wait` est spécifié, le statut final y sera enregistré.
- **`%neptune_ml export status`** : récupère le statut d'une tâche d'exportation.

  Parameters

  - **`--job-id`***`export job ID`*— L'ID de la tâche d'exportation dont le statut doit être récupéré.
  - **`--export-url`***`exporter-endpoint`*— (facultatif) Le point de terminaison Amazon API Gateway où l'exportateur peut être appelé.
  - **`--export-iam`** : (facultatif) indicateur signalant que les demandes adressées à l'URL d'exportation doivent être signées à l'aide de SigV4.
  - **`--export-no-ssl`** : (facultatif) indicateur signalant que le protocole SSL ne doit pas être utilisé lors de la connexion à l'exportateur.
  - **`--wait`** : (facultatif) indicateur signalant que l'opération doit attendre la fin de l'exportation.
  - **`--wait-interval`***`interval-to-wait`*— (facultatif) Définit le délai, en secondes, entre les vérifications du statut d'exportation (par défaut : 60).
  - **`--wait-timeout`***`timeout-seconds`*— (facultatif) Définit le temps, en secondes, à attendre que la tâche d'exportation soit terminée avant de renvoyer le statut le plus récent (par défaut : 3 600).

  - **`--store-to`***`location-to-store-result`*— (facultatif) Variable dans laquelle stocker le résultat de l'exportation. Si `--wait` est spécifié, le statut final y sera enregistré.
- **`%neptune_ml dataprocessing start`** : lance l'étape de traitement des données Neptune ML.

  Parameters

  - **`--job-id`***`ID for this job`*— (facultatif) ID à attribuer à cette tâche.
  - **`--s3-input-uri`***`S3 URI`*— (facultatif) L'URI S3 à laquelle trouver l'entrée pour cette tâche de traitement de données.
  - **`--config-file-name`***`file name`*— (facultatif) Nom du fichier de configuration pour cette tâche de traitement de données.
  - **`--store-to`***`location-to-store-result`*— (facultatif) La variable dans laquelle stocker le résultat du traitement des données.
  - **`--instance-type`***`(instance type)`*— (facultatif) Taille d'instance à utiliser pour cette tâche de traitement de données.
  - **`--wait`** : (facultatif) indicateur signalant que l'opération doit attendre la fin du traitement des données.
  - **`--wait-interval`***`interval-to-wait`*— (facultatif) Définit le délai, en secondes, entre les vérifications de l'état du traitement des données (par défaut : 60).
  - **`--wait-timeout`***`timeout-seconds`*— (facultatif) Définit le temps, en secondes, à attendre que la tâche de traitement des données soit terminée avant de renvoyer le statut le plus récent (par défaut : 3 600).
- **`%neptune_ml dataprocessing status`** : récupère le statut d'une tâche de traitement de données.

  Parameters

  - **`--job-id`***`ID of the job`*— ID de la tâche dont le statut doit être récupéré.
  - **`--store-to`***`instance type`*— (facultatif) Variable dans laquelle stocker le résultat de l'entraînement du modèle.
  - **`--wait`** : (facultatif) indicateur signalant que l'opération doit attendre la fin de l'entraînement du modèle.
  - **`--wait-interval`***`interval-to-wait`*— (facultatif) Définit le délai, en secondes, entre les vérifications de l'état d'entraînement du modèle (par défaut : 60).

  - **`--wait-timeout`***`timeout-seconds`*— (facultatif) Définit le temps, en secondes, à attendre que la tâche de traitement des données soit terminée avant de renvoyer le statut le plus récent (par défaut : 3 600).
- **`%neptune_ml training start`** : démarre le processus d'entraînement du modèle Neptune ML.

  Parameters

  - **`--job-id`***`ID for this job`*— (facultatif) ID à attribuer à cette tâche.
  - **`--data-processing-id`***`dataprocessing job ID`*— (facultatif) ID de la tâche de traitement des données qui a créé les artefacts à utiliser pour la formation.
  - **`--s3-output-uri`***`S3 URI`*— (facultatif) L'URI S3 à laquelle stocker le résultat de cette tâche d'entraînement de modèle.
  - **`--instance-type`***`(instance type)`*— (facultatif) Taille d'instance à utiliser pour cette tâche de formation de modèles.
  - **`--store-to`***`location-to-store-result`*— (facultatif) Variable dans laquelle stocker le résultat de l'entraînement du modèle.
  - **`--wait`** : (facultatif) indicateur signalant que l'opération doit attendre la fin de l'entraînement du modèle.
  - **`--wait-interval`***`interval-to-wait`*— (facultatif) Définit le délai, en secondes, entre les vérifications de l'état d'entraînement du modèle (par défaut : 60).
  - **`--wait-timeout`***`timeout-seconds`*— (facultatif) Définit le temps, en secondes, à attendre que la tâche de formation du modèle soit terminée avant de renvoyer le statut le plus récent (par défaut : 3 600).
- **`%neptune_ml training status`** : récupère le statut d'une tâche d'entraînement de modèle Neptune ML.

  Parameters

  - **`--job-id`***`ID of the job`*— ID de la tâche dont le statut doit être récupéré.
  - **`--store-to`***`instance type`*— (facultatif) Variable dans laquelle stocker le résultat du statut.
  - **`--wait`** : (facultatif) indicateur signalant que l'opération doit attendre la fin de l'entraînement du modèle.
  - **`--wait-interval`***`interval-to-wait`*— (facultatif) Définit le délai, en secondes, entre les vérifications de l'état d'entraînement du modèle (par défaut : 60).

  - **`--wait-timeout`***`timeout-seconds`*— (facultatif) Définit le temps, en secondes, à attendre que la tâche de traitement des données soit terminée avant de renvoyer le statut le plus récent (par défaut : 3 600).

- **`%neptune_ml endpoint create`** : crée un point de terminaison de requête pour un modèle Neptune ML.

  Parameters

  - **`--job-id`***`ID for this job`*— (facultatif) ID à attribuer à cette tâche.
  - **`--model-job-id`***`model-training job ID`*— (facultatif) ID de la tâche de formation de modèles pour laquelle créer un point de terminaison de requête.
  - **`--instance-type`***`(instance type)`*— (facultatif) Taille d'instance à utiliser pour le point de terminaison de la requête.
  - **`--store-to`***`location-to-store-result`*— (facultatif) Variable dans laquelle stocker le résultat de la création du point de terminaison.
  - **`--wait`** : (facultatif) indicateur signalant que l'opération doit attendre la fin de la création du point de terminaison.
  - **`--wait-interval`***`interval-to-wait`*— (facultatif) Définit le délai, en secondes, entre les vérifications de statut (par défaut : 60).
  - **`--wait-timeout`***`timeout-seconds`*— (facultatif) Définit le temps, en secondes, à attendre que le travail de création du point de terminaison soit terminé avant de renvoyer le statut le plus récent (par défaut : 3 600).

- **`%neptune_ml endpoint status`** : récupère le statut d'un point de terminaison de requête Neptune ML.

  Parameters

  - **`--job-id`***`endpoint creation ID`*— (facultatif) ID d'une tâche de création de point de terminaison dont le statut doit être signalé.
  - **`--store-to`***`location-to-store-result`*— (facultatif) Variable dans laquelle stocker le résultat du statut.
  - **`--wait`** : (facultatif) indicateur signalant que l'opération doit attendre la fin de la création du point de terminaison.
  - **`--wait-interval`***`interval-to-wait`*— (facultatif) Définit le délai, en secondes, entre les vérifications de statut (par défaut : 60).

- **`--wait-timeout`***`timeout-seconds`*— (facultatif) Définit le temps, en secondes, à attendre que le travail de création du point de terminaison soit terminé avant de renvoyer le statut le plus récent (par défaut : 3 600).

La magie cellulaire `%%neptune_ml`

La magie cellulaire `%%neptune_ml` ignore les entrées de ligne telles que `--job-id` ou `--export-url`. Au lieu de cela, elle vous permet de fournir ces entrées et d'autres au sein du corps de la cellule.

Vous pouvez également enregistrer ces entrées dans une autre cellule, attribuée à une variable Jupyter, puis les injecter dans le corps de la cellule à l'aide de cette variable. De cette façon, vous pouvez utiliser ces entrées encore et encore sans avoir à les saisir à chaque fois.

Cela ne fonctionne que si la variable d'injection est le seul contenu de la cellule. Vous ne pouvez pas utiliser plusieurs variables dans une cellule ni une combinaison de texte et de variable.

Par exemple, la magie cellulaire `%%neptune_ml export start` peut consommer un document JSON dans le corps de la cellule qui contient tous les paramètres décrits dans Paramètres utilisés pour contrôler le processus d'exportation de Neptune.

Dans le bloc-notes Neptune-ML-01-Introduction-to-Node-Classification-Gremlin, sous Configuration des fonctionnalités de la section Exportation de la configuration des données et du modèle, vous pouvez voir comment la cellule suivante détient les paramètres d'exportation dans un document attribué à une variable Jupyter nommée `export-params` :

```
export_params = {
  "command": "export-pg",
  "params": {
    "endpoint": neptune_ml.get_host(),
    "profile": "neptune_ml",
    "useIamAuth": neptune_ml.get_iam(),
    "cloneCluster": False
   },
  "outputS3Path": f'{s3_bucket_uri}/neptune-export',
  "additionalParams": {
    "neptune_ml": {
      "targets": [
        {
          "node": "movie",
```

```
          "property": "genre"
        }
      ],
      "features": [
        {
          "node": "movie",
          "property": "title",
          "type": "word2vec"
        },
        {
          "node": "user",
          "property": "age",
          "type": "bucket_numerical",
          "range" : [1, 100],
          "num_buckets": 10
        }
      ]
    }
  },
  "jobSize": "medium"}
```

Lorsque vous exécutez cette cellule, Jupyter enregistre le document de paramètres sous ce nom. Ensuite, vous pouvez utiliser `${export_params}` pour injecter le document JSON dans le corps d'une cellule `%%neptune_ml export start cell`, comme ceci :

```
%%neptune_ml export start --export-url {neptune_ml.get_export_service_host()} --export-
iam --wait --store-to export_results

${export_params}
```

Formes disponibles de la magie cellulaire **`%%neptune_ml`**

La magie cellulaire `%%neptune_ml` peut être utilisée sous les formes suivantes :

- **`%%neptune_ml export start`** : lance un processus d'exportation Neptune ML.
- **`%%neptune_ml dataprocessing start`** : démarre une tâche de traitement de données Neptune ML.
- **`%%neptune_ml training start`** : démarre une tâche d'entraînement de modèle Neptune ML.
- **`%%neptune_ml endpoint create`** : crée un point de terminaison de requête Neptune ML pour un modèle.

Visualisation des données graphiques dans les blocs-notes Amazon Neptune

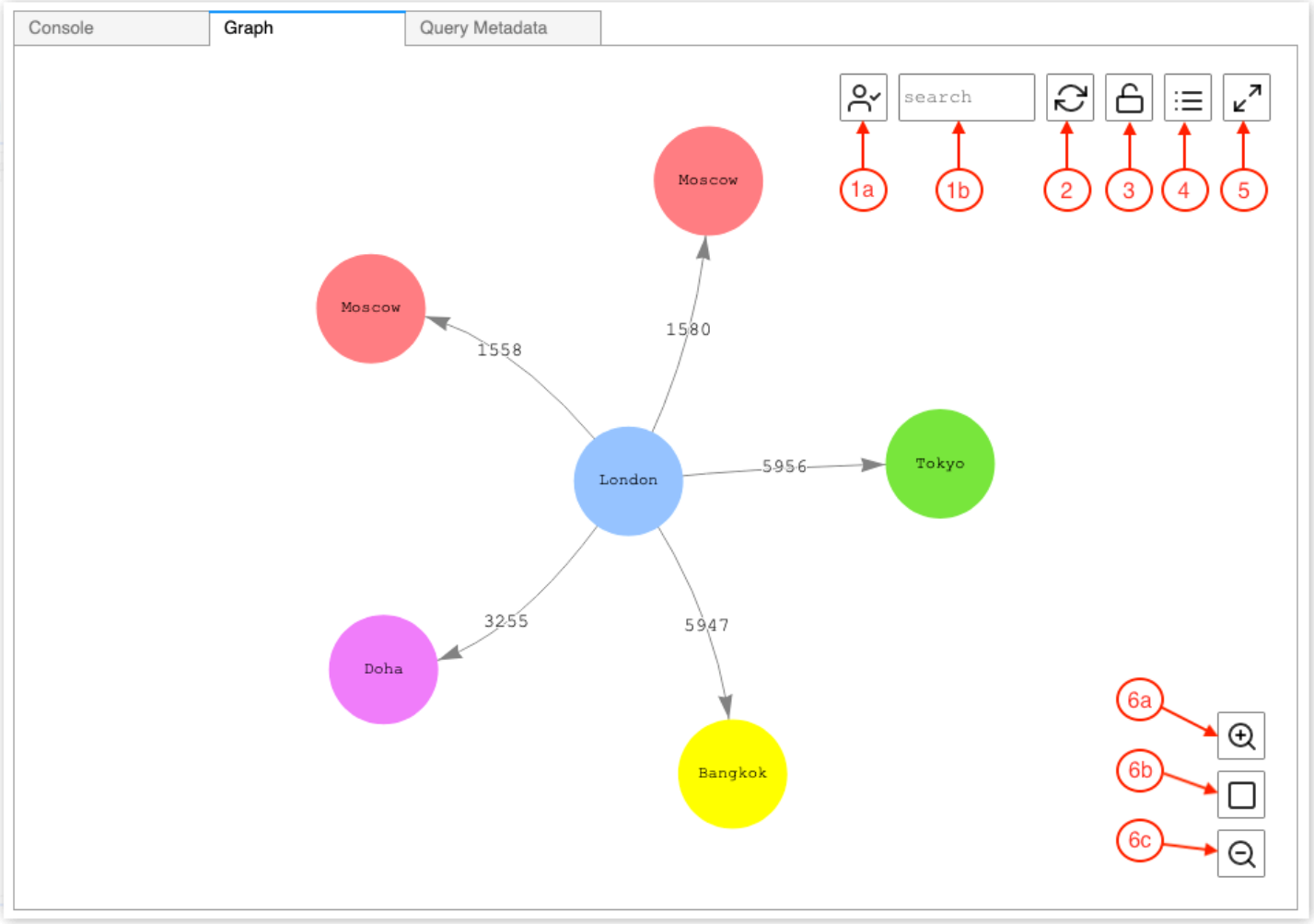
Dans de nombreux cas, le workbench Neptune peut créer un schéma visuel des résultats des requêtes et les renvoyer sous forme de tableau. La visualisation de graphe est disponible dans l'onglet Graphe des résultats de la requête chaque fois que la visualisation est possible.

Outre les fonctionnalités de visualisation intégrées décrites ici, vous pouvez également utiliser des outils de visualisation plus avancés avec les blocs-notes de graphe Neptune.

> **Note**
>
> Pour accéder aux fonctionnalités et aux correctifs récemment ajoutés aux blocs-notes que vous utilisez déjà, arrêtez d'abord votre instance de bloc-notes, puis redémarrez-la.

Exploration de l'interface Graph Tab dans Amazon Neptune

Ce schéma identifie les éléments d'interface utilisateur présents dans l'onglet Graphe :



1. Recherche de graphe
   a. Activation des UUID : active l'inclusion des valeurs des propriétés d'ID dans la recherche de graphe. Par défaut, l'inclusion d'ID est activée. Si cette option est désactivée, les correspondances sur les propriétés d'identification, y compris les propriétés de bord faisant référence à un nœud IDs, n'entraînent pas la mise en évidence des éléments.
   b. Champ de texte de recherche : met en évidence toutes les valeurs des propriétés des sommets et des arêtes qui contiennent la chaîne de texte que vous spécifiez ici.
2. Réinitialisation du graphe : réexécute la simulation de la physique du graphe et définit le zoom pour qu'il s'adapte au graphe dans la fenêtre.
3. Activer/désactiver la physique du graphe : active l'exécution de la simulation de la physique du graphe. La physique est activée par défaut, ce qui permet au graphe de changer dynamiquement. Si cette option est désactivée, les sommets restent verrouillés lorsque d'autres sommets sont déplacés.

4. Vue détaillée : lorsqu'un nœud ou une arête est sélectionné, une liste des clés et des valeurs de propriété de l'élément s'affiche, si elles sont disponibles dans les résultats de la requête.
5. Affichage plein écran : agrandit la fenêtre de l'onglet Graphe pour l'adapter à l'écran. Cliquez à nouveau pour réduire l'onglet Graphe.
6. Options de zoom
   a. Zoom avant
   b. Réinitialisation du zoom : définit le zoom pour adapter tous les sommets à la fenêtre de l'onglet Graphe.
   c. Zoom arrière

Visualisation des résultats de requêtes Gremlin

Le workbench Neptune crée une visualisation des résultats de requête pour toute requête Gremlin renvoyant un élément `path`. Pour voir la visualisation, sélectionnez l'onglet Graphe situé à droite de l'onglet Console sous la requête après l'avoir exécutée.

Vous pouvez utiliser les indicateurs de visualisation des requêtes pour contrôler la manière dont le visualiseur affiche les résultats des requêtes. Ces indicateurs respectent la magie cellulaire `%%gremlin` et sont précédés du nom de paramètre `--path-pattern` (ou de sa forme abrégée, `-p`) :

```
%%gremlin -p comma-separated hints
```

Vous pouvez également utiliser l'indicateur `--group-by` (ou `-g`) pour spécifier une propriété des sommets en fonction de laquelle les regrouper. Cela permet de spécifier une couleur ou une icône pour différents groupes de sommets.

Le nom des indicateurs reflète les étapes Gremlin couramment utilisées pour passer d'un sommet à l'autre. Ces indicateurs se comportent en conséquence. Plusieurs indicateurs peuvent être utilisés en combinaison, séparés par des virgules, sans aucun espace entre eux. Les indicateurs utilisés doivent correspondre aux étapes Gremlin correspondantes dans la requête visualisée. Voici un exemple :

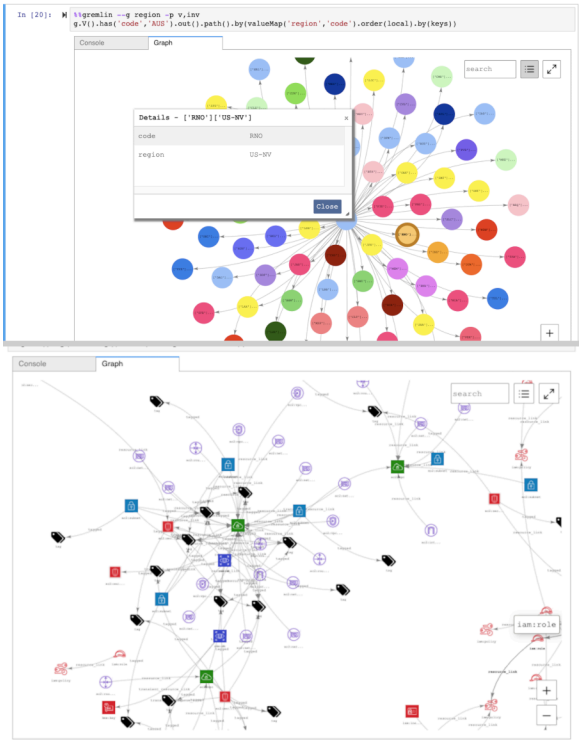
```
%%gremlin -p v,oute,inv
g.V().hasLabel('airport').outE().inV().path().by('code').by('dist').limit(5)
```

Voici les indicateurs de visualisation disponibles :

```
v
```

```
inv
outv
e
ine
oute
```

Voici quelques exemples de visualisations de graphes utilisant des groupes :



Visualisation des résultats de requêtes SPARQL

Le workbench Neptune crée une visualisation des résultats de requête pour toute requête SPARQL sous l'une des formes suivantes :

- `SELECT ?subject ?predicate ?object`
- `SELECT ?s ?p ?o`

Pour voir la visualisation, sélectionnez l'onglet Graphe situé à droite de l'onglet Table sous la requête après l'avoir exécutée.

Par défaut, une visualisation SPARQL inclut uniquement des modèles triples où `o?` est un `uri` ou un nœud vide (`bnode`). Tous les autres types de liaison `?o`, tels que les chaînes littérales ou les entiers,

sont traités comme des propriétés du nœud `?s` qui peuvent être visualisées dans le volet Details (Détails) de l'onglet Graph (Graphe).

Dans de nombreux cas, toutefois, vous souhaiterez peut-être inclure des valeurs littérales telles que les sommets dans la visualisation. Pour ce faire, utilisez l'indicateur de requête `--expand-all` situé après la magie cellulaire `%%sparql` :

```
%%sparql --expand-all
```

Cela indique au visualiseur d'inclure tous les résultats `?s ?p ?o` dans le schéma du graphe, quel que soit le type de liaison.

Vous pouvez voir cet indicateur utilisé dans tout le bloc-notes `Air-Routes-SPARQL.ipynb` et vous pouvez expérimenter en exécutant les requêtes avec ou sans cet indicateur pour voir quelle différence cela fait dans la visualisation.

Accès aux blocs-notes didactiques de visualisation dans le workbench Neptune

Les deux blocs-notes didactiques de visualisation fournis avec le workbench Neptune fournissent de nombreux exemples Gremlin et SPARQL sur la manière d'interroger efficacement les données de graphe et de visualiser les résultats.

Accès aux blocs-notes de visualisation

1. Dans le panneau de navigation à gauche, choisissez le bouton Ouvrir le bloc-notes à droite.
2. Une fois le workbench Neptune ouvert, en exécutant Jupyter, vous verrez un dossier Neptune au niveau supérieur. Choisissez-le pour ouvrir le dossier.
3. Au niveau suivant se trouve un dossier nommé 02-Visualization. Ouvrez ce dossier. À l'intérieur se trouvent plusieurs blocs-notes qui vous expliquent comment interroger les données de votre graphe, dans Gremlin et dans SPARQL, et comment visualiser les résultats des requêtes :

   - Air-Routes-Gremlin
   - Air-Routes-SPARQL
   - Blog sur la visualisation dans le workbench
   - EPL-Gremlin
   - EPL-SPARQL

Sélectionnez un bloc-notes pour tester les requêtes qu'il contient.

Résolution des problèmes liés aux blocs-notes Amazon Neptune

Lorsque vous travaillez avec des blocs-notes Neptune JupyterLab, vous pouvez parfois rencontrer des problèmes liés au comportement du navigateur, à JupyterLab lui-même ou à l'intégration d'Amazon SageMaker AI. Les sections suivantes décrivent les problèmes les plus courants et expliquent comment les résoudre.

Désynchronisation de l'état du widget

Les widgets Graph-Notebook peuvent parfois être désynchronisés entre l' JupyterLab interface utilisateur et le backend du noyau. Dans ce cas, vous pouvez voir des erreurs d'affichage du widget ou un message d'erreur comme celui-ci :

```
Model class 'OutputModel' from module '@jupyter-widgets/output' is loaded but cannot be
 instantiated
```

Cela se produit généralement après de longues périodes d'inactivité, lors du passage d'un onglet à un autre, après avoir vidé le cache du navigateur, lors d'une connectivité réseau instable ou lors de la réouverture d'un bloc-notes dont les sorties de widgets n'ont pas été enregistrées.

Pour résoudre ce problème, choisissez Redémarrer le noyau et effacer toutes les sorties dans le menu Kernel, puis réexécutez les cellules de votre bloc-notes.

JupyterLab 4.x retard de l'écran de chargement initial

Lorsque vous ouvrez un bloc-notes pour la première fois après le JupyterLab démarrage, un écran de chargement persistant peut s'afficher avec le message suivant : `Loading... The loading screen is taking a long time. Would you like to clear the workspace or keep waiting?`

Cela concerne principalement Firefox et ne se produit que sur le premier bloc-notes ouvert dans une session. Vous pouvez le résoudre en choisissant Effacer l'espace de travail, en actualisant la page du navigateur ou en rouvrant simplement le bloc-notes.

Si vous utilisez Firefox et que vous rencontrez ce problème à plusieurs reprises, essayez de supprimer les cookies et les données du site pour votre domaine SageMaker AI dans les paramètres de confidentialité et de sécurité de Firefox, puis actualisez la JupyterLab page.

Problèmes d'accès à Graph Explorer

Après le redémarrage du bloc-notes ou lorsque vous utilisez des informations d'identification mises en cache, vous pouvez rencontrer des difficultés pour accéder à Graph Explorer. Cela peut se traduire par des erreurs d'authentification, des messages d'accès refusé ou un échec du chargement complet de Graph Explorer.

Pour contourner ce problème, essayez de vider le cache et les cookies de votre navigateur. Vous pouvez également ouvrir JupyterLab Jupyter Classic d'abord pour forcer l'actualisation des informations d'identification, puis ouvrir Graph Explorer. Si le problème persiste, essayez d'utiliser une fenêtre de navigation privée ou privée.

Conseils généraux

De nombreux problèmes liés aux blocs-notes sont liés au navigateur. Pour minimiser les problèmes, tenez compte des points suivants :

- Videz régulièrement le cache de votre navigateur lorsque vous travaillez avec des blocs-notes Neptune.
- Utilisez toujours la session de bloc-notes la plus récente plutôt que de rouvrir les sessions périmées.
- Utilisez une fenêtre de navigation privée ou privée si vous rencontrez des problèmes persistants.
- Maintenez vos extensions JupyterLab et celles de Graph-Notebook à jour.

Configuration d'Amazon Neptune

Amazon Neptune est un service de base de données de graphes entièrement géré qui facilite la création et l'exécution d'applications fonctionnant avec des ensembles de données hautement connectés. Pour vous aider à démarrer avec Neptune, cette section de la documentation couvre un large éventail de sujets, tels que le choix des types d'instances et des configurations de stockage appropriés, la connexion sécurisée à votre cluster Neptune, le chargement des données et le suivi des performances. Les principaux domaines abordés dans cette section sont les suivants :

- Types d'instances : La documentation décrit les différents types d'instances disponibles pour Neptune, ainsi que leurs spécifications et considérations pour vous aider à sélectionner l'option la plus adaptée à votre charge de travail.
- Types de stockage : il couvre également les différents types de stockage pris en charge par Neptune et fournit des conseils sur la manière de choisir la bonne configuration de stockage.
- Création de clusters : La documentation fournit des instructions détaillées pour créer un nouveau cluster Neptune, notamment comment configurer les paramètres VPC requis.
- Connectivité et sécurité : il explique comment se connecter en toute sécurité à votre cluster Neptune, configurer l'accès VPC et configurer les mesures de sécurité nécessaires pour protéger vos données.
- Chargement des données : La documentation fournit des conseils sur le chargement de données dans votre cluster Neptune, y compris des options pour le chargement en masse et le streaming de données à l'aide de divers outils et API.
- Surveillance et résolution des problèmes : il contient également des informations sur la surveillance de votre environnement Neptune, l'accès aux journaux et la résolution des problèmes courants susceptibles de survenir pendant le fonctionnement.

> **Note**
>
> Pour les architectures de référence des bases de données AWS graphiques et les architectures de déploiement de référence, consultez Amazon Neptune Resources. Ces ressources peuvent vous aider dans vos choix de modèles de données de graphe et de langages de requête et vous permettront d'accélérer votre processus de développement.

Choix des types d'instances pour Amazon Neptune

Amazon Neptune propose différentes tailles et familles d'instances, qui offrent des fonctionnalités distinctes adaptées aux différentes charges de travail de graphe. Cette section a pour but de vous aider à choisir le type d'instance le plus adapté à vos besoins.

Pour connaître la tarification de chaque type d'instance dans ces familles, consultez la page de tarification de Neptune.

Présentation de l'allocation des ressources d'instance

Chaque type et taille d'instance Amazon EC2 utilisés dans Neptune offrent une quantité définie de calcul (vCPU) et de mémoire système. Le stockage principal de Neptune est externe aux instances de base de données d'un cluster, ce qui permet aux capacités de calcul et de stockage d'évoluer indépendamment l'une de l'autre.

Cette section se concentre sur la manière dont les ressources de calcul peuvent être mises à l'échelle et sur les différences entre les différentes familles d'instances.

Dans toutes les familles d'instances, des ressources vCPU sont allouées pour prendre en charge deux (2) threads d'exécution de requêtes par vCPU. Cette prise en charge est dictée par la taille de l'instance. Lorsque vous déterminez la taille appropriée d'une instance de base de données Neptune spécifique, vous devez tenir compte de la simultanéité possible de votre application et de la latence moyenne des requêtes. Vous pouvez estimer le nombre de vCPU nécessaires comme suit, où la latence est mesurée comme étant la latence moyenne des requêtes en secondes et la simultanéité comme correspondant au nombre cible de requêtes par seconde :

```
vCPUs = (latency x concurrency) / 2
```

> **Note**
>
> Les requêtes SPARQL, les requêtes openCypher et les requêtes de lecture Gremlin qui utilisent le moteur de requêtes DFE peuvent, dans certaines circonstances, utiliser plus d'un thread d'exécution par requête. Lors du dimensionnement initial du cluster de bases de données, partez du principe que chaque requête consomme un seul thread d'exécution par exécution, puis augmentez la taille si vous observez une pression de retour dans la file d'attente des requêtes. Cela peut être observé à l'aide des `/sparql/status`

API `/gremlin/status/oc/status`,, ou il peut également être observé à l'aide de la `MainRequestsPendingRequestsQueue` CloudWatch métrique.

La mémoire système de chaque instance est divisée en deux allocations principales : le cache du pool de tampons et la mémoire des threads d'exécution des requêtes.

Environ les deux tiers de la mémoire disponible dans une instance sont alloués au cache du pool de mémoire tampon. Buffer-pool le cache est utilisé pour mettre en cache les derniers composants du graphe utilisés afin d'accélérer l'accès aux requêtes qui accèdent à plusieurs reprises à ces composants. Les instances dotées d'une plus grande quantité de mémoire système possèdent de plus vastes caches de pool de tampons qui peuvent stocker une plus grande partie du graphe localement. Un utilisateur peut régler la quantité appropriée de cache du pool de mémoire tampon en surveillant les mesures de réussite et d'échec du cache tampon disponibles dans. CloudWatch

Il peut être utile d'augmenter la taille de l'instance si le taux d'accès au cache tombe en dessous de 99,9 % pendant une période constante. Cela suggère que le pool de tampons n'est pas assez grand et que le moteur doit récupérer les données du volume de stockage sous-jacent plus souvent que nécessaire.

Le tiers restant de la mémoire système est réparti uniformément entre les threads d'exécution des requêtes, alors qu'une partie de la mémoire reste allouée au système d'exploitation et à un petit pool dynamique pour les threads à utiliser selon les besoins. La mémoire disponible pour chaque thread augmente légèrement d'une taille d'instance à l'autre jusqu'à un type d'instance `8xl`, taille à laquelle la mémoire allouée par thread atteint un maximum.

Il convient d'ajouter de la mémoire de thread lorsque vous rencontrez une exception `OutOfMemoryException` (OOM). Les exceptions OOM se produisent lorsqu'un thread a besoin de plus de mémoire que la quantité maximale qui lui est allouée (ce qui diffère de la saturation de mémoire de l'instance entière).

types d'instances t3 et t4g

La famille d'instances `t3` et `t4g` offre une option économique pour commencer à utiliser une base de données orientée graphe, ainsi que pour le développement et les tests initiaux. Ces instances sont éligibles à l'offre gratuite de Neptune, qui permet aux nouveaux clients d'utiliser Neptune gratuitement pendant les 750 premières heures d'instance utilisées dans un AWS compte autonome ou regroupées auprès d'une AWS organisation avec facturation consolidée (compte payeur).

Les instances t3 et t4g ne sont proposées que dans les configurations de taille moyenne (t3.medium et t4g.medium).

Elles ne sont pas destinées à être utilisées dans un environnement de production.

Les ressources de ces instances étant très limitées, elles ne sont pas recommandées pour tester le temps d'exécution des requêtes ni les performances globales de la base de données. Pour évaluer les performances des requêtes, passez à une famille d'instances supérieure.

famille de types d'instances **r4**

OBSOLÈTE — La r4 gamme a été proposée lors du lancement de Neptune en 2018, mais les nouveaux types d'instances offrent désormais de bien meilleures performances. price/performance Depuis la version 1.1.0.0 du moteur, Neptune ne prend plus en charge les types d'instances r4.

famille de types d'instances **r5**

La famille r5 contient des types d'instances à mémoire optimisée qui sont adaptées à la plupart des cas d'utilisation de graphes. La famille r5 contient les types d'instances allant de r5.large jusqu'à r5.24xlarge. Les performances de calcul évoluent de manière linéaire à mesure que vous augmentez la taille. Par exemple, une instance r5.xlarge (4 vCPU et 32 Go de mémoire) possède deux fois plus de vCPU et de mémoire qu'une instance r5.large (2 vCPU et 16 Go de mémoire), et une instance r5.2xlarge (8 vCPU et 64 Go de mémoire) possède deux fois plus de vCPU et de mémoire qu'une instance r5.xlarge. Les performances des requêtes devraient évoluer directement en fonction de la capacité de calcul du type d'instance r5.12xlarge.

La famille d'instances r5 présente une architecture de processeur Intel à 2 sockets. Les instances r5.12xlarge et les types d'instances plus petits utilisent un seul socket et la mémoire système de ce processeur monosocket. Les types d'instances r5.16xlarge et r5.24xlarge utilisent à la fois les sockets et la mémoire disponible. Étant donné qu'une certaine surcharge de gestion de la mémoire est requise entre deux processeurs physiques dans une architecture à deux sockets, les gains de performances obtenus lors du passage d'une instance r5.12xlarge à un type d'instance r5.16xlarge ou r5.24xlarge plus grand ne sont pas aussi linéaires que ceux obtenus avec des tailles plus petites.

famille de types d'instances **r5d**

Neptune possède une fonctionnalité de cache de recherche qui contribue à améliorer les performances des requêtes qui doivent récupérer et renvoyer un grand nombre de valeurs de

propriétés et de littéraux. Cette fonctionnalité est principalement utilisée par les clients dont les requêtes doivent renvoyer de nombreux attributs. Le cache de recherche améliore les performances de ces requêtes en récupérant ces valeurs d'attribut localement plutôt que de les rechercher à plusieurs reprises dans le stockage indexé Neptune.

Le cache de recherche est implémenté à l'aide d'un volume NVMe-attached EBS sur un type d'`r5d`instance. Il est activé à l'aide du groupe de paramètres d'un cluster. Lorsque les données sont extraites du stockage indexé Neptune, les valeurs des propriétés et les littéraux RDF sont mis en cache dans ce volume NVMe.

Si vous n'avez pas besoin de la fonctionnalité de cache de recherche, utilisez un type d'instance `r5` standard plutôt qu'une instance `r5d`, pour éviter le coût plus élevé de `r5d`.

La famille `r5d` possède des types d'instances de la même taille que la famille `r5`, allant de `r5d.large` à `r5d.24xlarge`.

famille de types d'instances **`r6g`**

AWS a développé son propre ARM-based processeur appelé Graviton, qui offre de meilleurs résultats price/performance que les équivalents Intel et AMD. La famille `r6g` utilise le processeur Graviton2. Lors de nos tests, le processeur Graviton2 offre des performances supérieures de 10 à 20 % pour les requêtes graphiques OLTP-style (contraintes). OLAP-ishLes requêtes plus volumineuses peuvent toutefois être légèrement moins performantes avec les processeurs Graviton2 qu'avec les processeurs Intel en raison des performances de pagination mémoire légèrement moins performantes.

Il est également important de noter que la famille `r6g` possède une architecture à socket unique. Dès lors, les performances évoluent de manière linéaire en fonction de la capacité de calcul en passant d'un type d'instance `r6g.large` à `r6g.16xlarge` (qui est le type d'instance le plus vaste dans cette famille).

famille de types d'instances **`r6i`**

Les instances Amazon R6i sont alimentées par des processeurs Intel Xeon Scalable de 3e génération (nom de code Ice Lake) et conviennent parfaitement aux charges de travail gourmandes en mémoire. En règle générale, elles offrent des performances de calcul jusqu'à 15 % supérieures et une bande passante mémoire par vCPU jusqu'à 20 % supérieure à celle des types d'instances R5 comparables.

famille de types d'instances **x2g**

Certains cas d'utilisation de graphes impliquent de meilleures performances lorsque les instances disposent de caches de pool de tampons plus grands. La famille `x2g` a été créée pour mieux prendre en charge ces cas d'utilisation. La famille `x2g` possède un ratio mémoire/vCPU supérieur à celui de la famille `r5` ou `r6g`. Les instances `x2g` utilisent également le processeur Graviton2 et ont de nombreuses caractéristiques de performance identiques à celles des types d'instances `r6g`, ainsi qu'un cache de pool de tampons plus grand.

Si vous utilisez des types `r5` d'`r6g`instance présentant une faible utilisation du processeur et un taux d'échec élevé du cache du pool de mémoire tampon, essayez plutôt d'utiliser la `x2g` famille. Ainsi, vous obtiendrez la mémoire supplémentaire dont vous avez besoin sans avoir à payer pour augmenter la capacité du processeur.

famille de types d'instances **x2iezn**

La `x2iezn` gamme fournit des instances optimisées pour la mémoire alimentées par des processeurs Intel Xeon Scalable dotés de performances à haute fréquence. Ces instances offrent un ratio mémoire/processeur virtuel élevé (32 GiB par vCPU), ce qui les rend parfaitement adaptées aux charges de travail graphiques gourmandes en mémoire qui bénéficient de performances élevées en mode monothread.

Les principales caractéristiques incluent une fréquence turbo tout-cœur allant jusqu'à 4,5 GHz et la disponibilité dans des tailles allant de 2 x large à 12 x large.

famille de types d'**instances x2iedn**

La `x2iedn` gamme fournit des instances optimisées pour la mémoire avec un stockage SSD NVMe local. Ces instances associent une capacité de mémoire élevée (32 GiB par vCPU) à un stockage local rapide, ce qui les rend idéales pour les charges de travail graphiques qui bénéficient à la fois de grands caches en mémoire et d'une mise en cache de disque locale à hautes performances.

Alimentées par des processeurs Intel Xeon Scalable de 3e génération, ces instances sont disponibles dans des tailles allant de xlarge à 32xlarge et sont optimisées pour les bases de données graphiques à grande échelle nécessitant à la fois des performances de mémoire et de stockage.

famille de types d'instances **r8g**

La `r8g` famille comprend des types d'instances optimisés pour la mémoire alimentés par des processeurs AWS Graviton4. Ces instances offrent des améliorations de performances significatives

par rapport aux générations précédentes, ce qui les rend parfaitement adaptées aux charges de travail graphiques gourmandes en mémoire. Les instances r8g offrent des performances supérieures d'environ 15 à 20 % pour les requêtes graphiques par rapport aux instances r7g.

La `r8g` famille utilise une plate-forme à double socket. Les types d'instance vont de `r8g.large` à `r8g.24xlarge` s'exécuter sur un seul socket, ce qui signifie que les performances évoluent de manière linéaire en fonction de la capacité de calcul sur cette plage. Elle `r8g.48xlarge` utilise les deux sockets et constitue le type d'instance le plus important de la famille ; comme les autres familles à double socket, les gains de performances lors du passage de `r8g.24xlarge` à `r8g.48xlarge` peuvent ne pas être parfaitement linéaires en raison de la surcharge de gestion de la mémoire entre sockets.

Les principales caractéristiques de la `r8g` famille sont les suivantes :

- Alimenté par les processeurs AWS Graviton4 ARM-based
- Bande passante mémoire supérieure par vCPU par rapport aux générations précédentes
- Excellent price/performance ratio pour les requêtes graphiques OLTP-style (contraintes) et les charges de travail OLAP-style analytiques
- Fonctionnalités de gestion de la mémoire améliorées qui favorisent les traversées de graphes complexes

La `r8g` gamme est idéale pour les charges de travail de production qui nécessitent une capacité de mémoire élevée et des performances constantes. Ils sont particulièrement efficaces pour les applications soumises à des exigences élevées en matière de simultanéité des requêtes.

famille de types d'instances **`r7g`**

La `r7g` famille utilise le processeur AWS Graviton3, qui offre de meilleurs résultats price/performance que les instances précédentes Graviton2-based . Lors des tests, le processeur Graviton3 offre des performances de 25 à 30 % supérieures de 25 à 30 % pour les requêtes OLTP-style graphiques par rapport aux instances r6g.

À l'instar de la `r6g` famille, la `r7g` famille possède une architecture à socket unique, ce qui signifie que les performances évoluent de manière linéaire en fonction de la capacité de calcul comprise entre un `r7g.large` et un `r7g.16xlarge` (le type le plus important de la famille).

Les principales caractéristiques de la `r7g` famille sont les suivantes :

- Alimenté par les processeurs AWS Graviton3 ARM-based
- Performances de pagination mémoire améliorées par rapport au r6g, ce qui profite à la fois aux charges de travail OLTP et OLAP
- Efficacité améliorée du cache du pool de mémoire tampon
- Latence réduite pour les opérations gourmandes en mémoire

La `r7g` gamme convient parfaitement aux environnements de production présentant des modèles de requêtes variés et est particulièrement efficace pour les charges de travail qui bénéficient d'une bande passante mémoire améliorée.

famille de types d'instances **`r7i`**

La `r7i` famille est alimentée par des processeurs Intel Xeon Scalable de 4e génération (nom de code Sapphire Rapids) et offre des améliorations significatives par rapport aux instances r6i. Ces instances fournissent un calcul supérieur d'environ 15 % price/performance et une bande passante mémoire jusqu'à 20 % supérieure par vCPU par rapport aux types d'instances r6i comparables.

La famille d'`r7i`instances possède une architecture de processeur Intel à 2 sockets, similaire à la `r5` famille. Les instances `r7i.12xlarge` et les types d'instances plus petits utilisent un seul socket et la mémoire système de ce processeur monosocket. Les types d'instances `r7i.16xlarge` et `r7i.24xlarge` utilisent à la fois les sockets et la mémoire disponible. Étant donné qu'une certaine surcharge de gestion de la mémoire est requise entre deux processeurs physiques dans une architecture à deux sockets, les gains de performances obtenus lors du passage d'une instance `r7i.12xlarge` à un type d'instance `r7i.16xlarge` ou `r7i.24xlarge` plus grand ne sont pas aussi linéaires que ceux obtenus avec des tailles plus petites.

Les principales caractéristiques de la `r7i` famille sont les suivantes :

- Alimenté par des processeurs Intel Xeon Scalable de 4e génération
- Les performances évoluent de manière linéaire avec une capacité de calcul allant jusqu'à r7i.12xlarge
- Gestion améliorée de la mémoire entre les processeurs physiques dans l'architecture à 2 sockets
- Performances améliorées pour les opérations graphiques gourmandes en mémoire

Pour toutes ces familles d'instances, vous pouvez estimer le nombre de vCPU nécessaires en utilisant la même formule que celle mentionnée précédemment :

```
vCPUs = (latency x concurrency) / 2
```

Où la latence est mesurée comme la latence moyenne des requêtes en secondes et la simultanéité comme le nombre cible de requêtes par seconde.

type d'instance **sans serveur**

La fonctionnalité Neptune sans serveur permet de mettre à l'échelle la taille de l'instance de manière dynamique en fonction des besoins en ressources d'une charge de travail. Au lieu de calculer le nombre de vCPU nécessaires à votre application, Neptune sans serveur vous permet de définir des limites inférieures et supérieures de capacité de calcul (mesurées en unités de capacité Neptune) pour les instances de votre cluster de bases de données. Les charges de travail dont l'utilisation varie peuvent être optimisées en termes de coûts en utilisant des instances sans serveur plutôt que des instances provisionnées.

Vous pouvez configurer à la fois des instances provisionnées et des instances sans serveur dans le même cluster de bases de données pour obtenir une configuration coût/performance optimale.

Choix des types de stockage pour Amazon Neptune

Neptune propose deux types de stockage avec un modèle de tarification différent :

- Stockage standard : le stockage standard fournit un stockage de base de données économique pour les applications dont l' I/O utilisation est modérée à faible.
- I/O—Stockage optimisé — Avec I/O —Optimized Storage, disponible à partir de la version 1.3.0.0 du moteur, vous ne payez que pour le stockage et les instances que vous utilisez. Les coûts de stockage sont plus élevés que pour le stockage standard, et les coûts d'instance sont également plus élevés que pour les instances standard. Vous ne payez rien pour I/O ce que vous utilisez. Si votre I/O utilisation est élevée, le stockage IOPS provisionné peut réduire les coûts de manière significative.

  I/O—Le stockage optimisé est conçu pour répondre aux besoins des charges de travail graphiques I/O intensives à un coût prévisible. Vous ne pouvez passer du type de stockage I/O optimisé au type de stockage standard qu'une fois tous les 30 jours.

  Pour obtenir des informations sur les tarifs I/O relatifs au stockage optimisé, consultez la page de tarification de Neptune. La section suivante décrit comment configurer le stockage I/O optimisé pour un cluster de base de données Neptune.

Choix I/O : stockage optimisé pour un cluster de base de données Neptune

Par défaut, les clusters de base de données Neptune utilisent le stockage standard. Vous pouvez activer le stockage I/O optimisé sur un cluster de base de données au moment de sa création, comme suit :

Voici un exemple de la façon dont vous pouvez activer le I/O stockage optimisé lorsque vous créez un cluster à l'aide de AWS CLI :

```
aws neptune create-db-cluster \
  --db-cluster-identifier (an ID for the cluster) \
  --engine neptune \
  --engine-version (the Neptune engine version) \
  --storage-type iopt1
```

Ensuite, le stockage I/O optimisé est activé sur toute instance que vous créez automatiquement :

```
aws neptune create-db-instance \
  --db-cluster-identifier (the ID of the new cluster) \
  --db-instance-identifier (an ID for the new instance) \
  --engine neptune \
  --db-instance-class db.r5.large
```

Vous pouvez également modifier un cluster de base de données existant pour I/O y activer le stockage optimisé, comme suit :

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (the ID of a cluster without I/O-Optimized storage) \
  --storage-type iopt1 \
  --apply-immediately
```

Vous pouvez restaurer un instantané de sauvegarde sur un cluster de base de données avec I/O — Optimized storage activé :

```
aws neptune restore-db-cluster-from-snapshot \
  --db-cluster-identifier (an ID for the restored cluster) \
  --snapshot-identifier (the ID of the snapshot to restore from) \
  --engine neptune \
  --engine-version (the Neptune engine version) \
  --storage-type iopt1
```

Vous pouvez déterminer si un cluster utilise un stockage I/O optimisé à l'aide d'un `describe-` appel. Si le stockage I/O optimisé est activé, l'appel renvoie un champ de type de stockage défini sur. `iop1`

Création d'un cluster Amazon Neptune

Le moyen le plus simple de créer un nouveau cluster de base de données Amazon Neptune consiste à utiliser un CloudFormation modèle qui crée toutes les ressources nécessaires pour vous, sans avoir à tout faire manuellement. Le CloudFormation modèle effectue une grande partie de la configuration à votre place, notamment la création d'une instance Amazon Elastic Compute Cloud (Amazon EC2) :

Pour lancer un nouveau cluster de base de données Neptune à l'aide d'un CloudFormation modèle

1. Créez un utilisateur IAM disposant des autorisations dont vous avez besoin pour travailler avec votre cluster de bases de données Neptune, comme expliqué dans Autorisations des utilisateurs IAM.
2. Définissez les prérequis supplémentaires nécessaires pour utiliser le CloudFormation modèle, comme expliqué dansConditions préalables à la configuration d'Amazon Neptune à l'aide de AWS CloudFormation.
3. Invoquez la CloudFormation pile, comme décrit dansCréation d'un cluster Amazon Neptune à l'aide de AWS CloudFormation.

Vous pouvez également créer une base de données globale Neptune qui couvre plusieurs bases de données Régions AWS, ce qui permet des lectures globales à faible latence et une restauration rapide dans les rares cas où une panne affecte l'ensemble d'une base de données. Région AWS

Pour plus d'informations sur la création manuelle d'un cluster Amazon Neptune à l'aide du AWS Management Console, consultez. Lancement d'un cluster de base de données Neptune à l'aide du AWS Management Console

Vous pouvez également utiliser un CloudFormation modèle pour créer une fonction Lambda à utiliser avec Neptune (voir). Utilisation CloudFormation pour créer une fonction Lambda à utiliser dans Neptune

Pour plus d'informations sur la gestion des clusters et des instances dans Neptune, consultez Gestion de votre base de données Amazon Neptune.

Conditions préalables à la configuration d'Amazon Neptune à l'aide de AWS CloudFormation

Avant de créer un cluster Amazon Neptune à l'aide d'un CloudFormation modèle, vous devez disposer des éléments suivants :

- Une paire de clés Amazon EC2
- Les autorisations requises pour l'utilisation CloudFormation.

Créez une paire de clés Amazon EC2 à utiliser pour lancer un cluster Neptune à l'aide de CloudFormation

Pour lancer un cluster de base de données Neptune à l'aide d'un CloudFormation modèle, vous devez disposer d'une paire Amazon EC2Key (et du fichier PEM associé) dans la région où vous créez la pile. CloudFormation

Si vous devez créer la paire de clés, consultez Création d'une paire de clés à l'aide d'Amazon EC2 dans le guide de l'utilisateur Amazon EC2 ou Création d'une paire de clés à l'aide d'Amazon EC2 dans le guide de l'utilisateur Amazon EC2 pour obtenir des instructions.

Ajoutez des politiques IAM pour accorder les autorisations nécessaires à l'utilisation du CloudFormation modèle

Tout d'abord, configurez un utilisateur IAM doté des autorisations nécessaires pour travailler avec Neptune, comme décrit dans Création d'un utilisateur IAM avec les autorisations nécessaires pour accéder à Neptune.

Vous devez ensuite ajouter la politique AWS gérée `AWSCloudFormationReadOnlyAccess` à cet utilisateur.

Enfin, créez la politique gérée par le client suivante et ajoutez-la à cet utilisateur :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/*",
            "Condition": {
                "StringEquals": {
                    "iam:passedToService": "rds.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "rds.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics",
                "sns:ListSubscriptions",
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:*:111122223333:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:ListRetirableGrants",
                "kms:ListKeys",
                "kms:ListAliases",
                "kms:ListKeyPolicies"
            ],
            "Resource": "arn:aws:kms:*:111122223333:key/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
```

```
            "Resource": "arn:aws:cloudwatch:*:111122223333:service/*-*",
            "Condition": {
                "StringLike": {
                    "cloudwatch:namespace": "AWS/Neptune"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeVpcs",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute"
            ],
            "Resource": [
                "arn:aws:ec2:*:111122223333:vpc/*",
                "arn:aws:ec2:*:111122223333:subnet/*",
                "arn:aws:ec2:*:111122223333:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBCluster",
                "rds:CreateDBInstance",
                "rds:AddTagsToResource",
                "rds:ListTagsForResource",
                "rds:RemoveTagsFromResource",
                "rds:RemoveRoleFromDBCluster",
                "rds:ResetDBParameterGroup",
                "rds:CreateDBSubnetGroup",
                "rds:ModifyDBParameterGroup",
                "rds:DownloadDBLogFilePortion",
                "rds:CopyDBParameterGroup",
                "rds:AddRoleToDBCluster",
                "rds:ModifyDBInstance",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:DeleteDBInstance",
                "rds:CopyDBClusterParameterGroup",
                "rds:CreateDBParameterGroup",
```

```
                "rds:DescribeDBSecurityGroups",
                "rds:DeleteDBSubnetGroup",
                "rds:DescribeValidDBInstanceModifications",
                "rds:ModifyDBCluster",
                "rds:CreateDBClusterSnapshot",
                "rds:DeleteDBParameterGroup",
                "rds:CreateDBClusterParameterGroup",
                "rds:RemoveTagsFromResource",
                "rds:PromoteReadReplicaDBCluster",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribePendingMaintenanceActions",
                "rds:DescribeDBParameterGroups",
                "rds:FailoverDBCluster",
                "rds:DescribeDBInstances",
                "rds:DescribeDBParameters",
                "rds:DeleteDBCluster",
                "rds:ResetDBClusterParameterGroup",
                "rds:RestoreDBClusterToPointInTime",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:AddTagsToResource",
                "rds:DescribeDBClusterParameters",
                "rds:CopyDBClusterSnapshot",
                "rds:DescribeDBLogFiles",
                "rds:DeleteDBClusterSnapshot",
                "rds:ListTagsForResource",
                "rds:RebootDBInstance",
                "rds:DescribeDBClusterSnapshots",
                "rds:DeleteDBClusterParameterGroup",
                "rds:ApplyPendingMaintenanceAction",
                "rds:DescribeDBClusters",
                "rds:DescribeDBClusterParameterGroups",
                "rds:ModifyDBSubnetGroup"
            ],
            "Resource": [
                "arn:aws:rds:*:111122223333:cluster-snapshot:*",
                "arn:aws:rds:*:111122223333:cluster:*",
                "arn:aws:rds:*:111122223333:pg:*",
                "arn:aws:rds:*:111122223333:cluster-pg:*",
                "arn:aws:rds:*:111122223333:secgrp:*",
                "arn:aws:rds:*:111122223333:db:*",
                "arn:aws:rds:*:111122223333:subgrp:*"
            ],
            "Condition": {
```

```
                "StringEquals": {
                    "rds:DatabaseEngine": [
                        "graphdb",
                        "neptune"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:111122223333:log-group:*:log-stream:*",
                "arn:aws:logs:*:111122223333:log-group:*"
            ]
        }
    ]
}
```

> **Note**
>
> Les autorisations suivantes sont uniquement requises pour supprimer une pile : `iam:DeleteRole`, `iam:RemoveRoleFromInstanceProfile`, `iam:DeleteRolePolicy`, `iam:DeleteInstanceProfile` et `ec2:DeleteVpcEndpoints`.
> Notez également que `ec2:*Vpc` accorde des autorisations `ec2:DeleteVpc`.

Création d'un cluster Amazon Neptune à l'aide de AWS CloudFormation

Vous pouvez utiliser un CloudFormation modèle pour configurer un cluster de base de données Neptune.

1. Pour lancer la CloudFormation pile sur la CloudFormation console, cliquez sur l'un des boutons Launch Stack dans le tableau suivant.

Région	Afficher	Afficher dans Designer	Lancer
USA Est (Virginie du Nord)	Afficher	Afficher dans Designer	
USA Est (Ohio)	Afficher	Afficher dans Designer	
USA Ouest (Californie du Nord)	Afficher	Afficher dans Designer	
USA Ouest (Oregon)	Afficher	Afficher dans Designer	
Canada (Centre)	Afficher	Afficher dans Designer	
Canada-Ouest (Calgary)	Afficher	Afficher dans Designer	
Amérique du Sud (São Paulo)	Afficher	Afficher dans Designer	
Europe (Stockholm)	Afficher	Afficher dans Designer	
Europe (Irlande)	Afficher	Afficher dans Designer	

Région	Afficher	Afficher dans Designer	Lancer
Europe (Londres)	Afficher	Afficher dans Designer	
Europe (Paris)	Afficher	Afficher dans Designer	
Europe (Espagne)	Afficher	Afficher dans Designer	
Europe (Francfort)	Afficher	Afficher dans Designer	
Europe (Zurich)	Afficher	Afficher dans Designer	
Middle East (Bahrain)	Afficher	Afficher dans Designer	
Moyen-Orient (EAU)	Afficher	Afficher dans Designer	
Israël (Tel Aviv)	Afficher	Afficher dans Designer	
Afrique (Le Cap)	Afficher	Afficher dans Designer	
Asie-Pacifique (Hong Kong)	Afficher	Afficher dans Designer	
Asie-Pacifique (Tokyo)	Afficher	Afficher dans Designer	
Asie-Pacifique (Séoul)	Afficher	Afficher dans Designer	

Région	Afficher	Afficher dans Designer	Lancer
Asie-Pacifique (Singapour)	Afficher	Afficher dans Designer	Launch Stack
Asie-Pacifique (Sydney)	Afficher	Afficher dans Designer	Launch Stack
Asie-Pacifique (Jakarta)	Afficher	Afficher dans Designer	Launch Stack
Asie-Pacifique (Melbourne)	Afficher	Afficher dans Designer	Launch Stack
Asie-Pacifique (Malaisie)	Afficher	Afficher dans Designer	Launch Stack
Asie-Pacifique (Mumbai)	Afficher	Afficher dans Designer	Launch Stack
Asie-Pacifique (Hyderabad)	Afficher	Afficher dans Designer	Launch Stack
Chine (Pékin)	Afficher	Afficher dans Designer	Launch Stack
Chine (Ningxia)	Afficher	Afficher dans Designer	Launch Stack
AWS GovCloud (US-West)	Afficher	Afficher dans Designer	Launch Stack
AWS GovCloud (US-East)	Afficher	Afficher dans Designer	Launch Stack

2. Sur la page Select Template, choisissez Next.
3. Sur la page Spécifier les détails, choisissez une paire de clés pour EC2SSHKeyPairName.

Cette paire de clés est requise pour accéder à l'instance EC2. Vérifiez que vous disposez du fichier PEM pour la paire de clés que vous choisissez.

4. Choisissez Suivant.
5. Dans la page Options, choisissez Suivant.
6. Sur la page Vérification, cochez la première case pour accepter que CloudFormation créera des ressources IAM. Cochez la deuxième case pour confirmer `CAPABILITY_AUTO_EXPAND` comme nouvelle pile.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` accepte explicitement que les macros soient étendues lors de la création de la pile, sans révision préalable. Les utilisateurs créent souvent un jeu de modifications à partir d'un modèle traité, de sorte que les modifications apportées par les macros puissent être révisées avant la création de la pile. Pour en savoir plus, consultez l'API CloudFormation CreateStack.

   Ensuite, choisissez Créer.

> **Note**
>
> Vous pouvez également utiliser votre CloudFormation modèle pour mettre à niveau la version du moteur de votre cluster de bases de données.

Configureation du VPC Amazon où se trouve le cluster de bases de données Amazon Neptune

Les clusters de bases de données Amazon Neptune peuvent uniquement être créés dans un Amazon Virtual Private Cloud (Amazon VPC). Ses points de terminaison sont accessibles au sein de ce VPC, et si les points de terminaison publics Neptune sont activés, ils sont également accessibles en dehors du VPC et via Internet.

Il existe différentes manières de configurer le VPC, en fonction de la manière dont vous souhaitez accéder à votre cluster de bases de données.

Voici quelques points à prendre en compte lors de la configuration du VPC sur lequel se trouve le cluster de bases de données Neptune :

- Votre VPC doit avoir au moins deux sous-réseaux. Ces sous-réseaux doivent être dans deux zones de disponibilité (AZ) différentes. En répartissant vos instances de cluster sur deux zones de disponibilité au moins, vous garantissez que des instances seront disponibles dans le cluster de bases de données, dans l'éventualité peu probable d'une défaillance d'une zone de disponibilité. Le volume de votre cluster de bases de données Neptune couvre toujours trois zones de disponibilité afin de fournir un stockage durable avec un risque extrêmement faible de perte de données.
- Les blocs d'adresse CIDR de chaque sous-réseau doivent être assez vastes pour fournir les adresses IP dont Neptune pourrait avoir besoin pendant les activités de maintenance, le basculement et la mise à l'échelle
- Le VPC doit avoir un groupe de sous-réseaux de base de données qui contient les sous-réseaux que vous avez créés. Neptune choisit l'un des sous-réseaux du groupe de sous-réseaux et une adresse IP dans ce sous-réseau à associer à chaque instance de base de données du cluster de bases de données. L'instance de base de données est donc située dans la même zone de disponibilité que le sous-réseau.
- Le DNS doit être activé (noms d'hôte DNS et résolution DNS) sur le VPC.
- Votre VPC doit avoir un groupe de sécurité VPC qui autorise l'accès au cluster de bases de données.
- La location dans un VPC Neptune doit être définie sur la valeur Par défaut.

Ajout de sous-réseaux au VPC où se trouve le cluster de bases de données Neptune

Un sous-réseau est une plage d'adresses IP dans votre VPC. Vous pouvez lancer des ressources telles qu'un cluster de bases de données Neptune ou une instance EC2 dans un sous-réseau spécifique. Lorsque vous créez un sous-réseau, vous spécifiez son bloc d'adresse CIDR IPv4, lequel est un sous-ensemble du bloc d'adresse CIDR du VPC. Chaque sous-réseau doit résider entièrement dans une seule zone de disponibilité (AZ) et ne peut pas couvrir plusieurs zones. En lançant des instances dans des zones de disponibilité distinctes, vous protégez vos applications contre toute défaillance éventuelle d'une de ces zones. Pour plus d'informations, consultez la documentation sur les sous-réseaux VPC.

Un cluster de bases de données Neptune nécessite au moins deux sous-réseaux VPC.

Pour ajouter des sous-réseaux à un VPC

1. Connectez-vous à la console Amazon VPC AWS Management Console et ouvrez-la à l'adresse. https://console.aws.amazon.com/vpc/
2. Dans le panneau de navigation, choisissez Subnets (Sous-réseaux).
3. Dans le coin supérieur gauche, sélectionnez Tableau de bord VPC, choisissez Sous-réseaux, puis Créer un sous-réseau.
4. Sur la page Créer un sous-réseau, choisissez le VPC dans lequel vous souhaitez créer le sous-réseau.
5. Sous Paramètres du sous-réseau, effectuez les choix suivants :
    a. Dans Nom du sous-réseau, indiquez un nom pour le nouveau sous-réseau.
    b. Choisissez une zone de disponibilité (AZ) pour ce sous-réseau ou conservez l'option Aucune préférence.
    c. Sous Bloc d'adresse CIDR IPv4, entrez le bloc d'adresse IP du sous-réseau.
    d. Ajoutez des balises au sous-réseau si nécessaire.
    e. Choix
6. Si vous souhaitez créer un autre sous-réseau en même temps, choisissez Ajouter un nouveau sous-réseau.
7. Choisissez Créer un sous -réseau pour créer un ou plusieurs autres sous-réseaux.

Configuration d'un VPC dans Amazon Neptune

Créez un groupe de sous-réseaux.

Pour créer un groupe de sous-réseaux

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Sélectionnez Groupes de sous-réseaux, puis Créer un groupe de sous-réseaux DB.
3. Saisissez un nom et une description pour le nouveau groupe de sous-réseaux (la description est obligatoire).
4. Sous VPC, choisissez le VPC dans lequel vous souhaitez que ce groupe de sous-réseaux soit situé.

5. Sous Zone de disponibilité, choisissez la zone de disponibilité dans laquelle vous souhaitez que ce groupe de sous-réseaux soit situé.
6. Sous Sous-réseau, ajoutez un ou plusieurs sous-réseaux de cette zone de disponibilité à ce groupe de sous-réseaux.
7. Choisissez Créer pour créer le groupe de sous-réseaux.

Créez un groupe de sécurité à l'aide de la console VPC.

Les groupes de sécurité autorisent l'accès au cluster de bases de données dans le VPC. Ils font office de pare-feu pour le cluster de bases de données associé, en contrôlant le trafic entrant et le trafic sortant au niveau de l'instance. Par défaut, une instance de base de données est créée avec un pare-feu et un groupe de sécurité par défaut qui en empêche l'accès. Pour activer l'accès, vous devez disposer d'un groupe de sécurité VPC doté de règles supplémentaires.

La procédure suivante vous montre comment ajouter une règle TCP personnalisée qui spécifie la plage de ports et les adresses IP que l'instance Amazon EC2 utilise pour accéder au cluster de bases de données Neptune. Vous pouvez utiliser le groupe de sécurité VPC attribué à l'instance EC2 au lieu de son adresse IP.

Pour créer un groupe de sécurité VPC pour Neptune sur la console

1. Connectez-vous à la console Amazon VPC AWS Management Console et ouvrez-la à l'adresse. https://console.aws.amazon.com/vpc/
2. Dans le coin supérieur droit de la console, choisissez la AWS région dans laquelle vous souhaitez créer un groupe de sécurité VPC pour Neptune. La liste des ressources Amazon VPC de cette région devrait indiquer que vous possédez au moins un VPC et plusieurs sous-réseaux. Si ce n'est pas le cas, cela signifie que vous ne disposez pas de VPC par défaut dans cette région.
3. Dans le volet de navigation, sous Sécurité, choisissez Groupes de sécurité.
4. Sélectionnez Create security group (Créer un groupe de sécurité). Dans la fenêtre Créer un groupe de sécurité, entrez le nom du groupe de sécurité, une description et l'identifiant du VPC dans lequel résidera le cluster de bases de données Neptune.
5. Ajoutez une règle entrante pour le groupe de sécurité d'une instance Amazon EC2 que vous souhaitez connecter au cluster de bases de données Neptune :

    a. Dans la zone Règles entrantes, choisissez Ajouter une règle.

b. Dans la liste Type, laissez l'option TCP personnalisé sélectionnée.

c. Dans la zone de texte Plage de ports, saisissez 8182, qui est la valeur de port par défaut pour Neptune.

d. Sous Source, entrez la plage d'adresses IP (valeur CIDR) à partir de laquelle vous allez accéder à Neptune, ou choisissez le nom d'un groupe de sécurité existant.

e. Si vous devez ajouter d'autres adresses IP ou des plages de ports différentes, choisissez à nouveau Ajouter une règle.

6. Dans la zone Règles sortantes, vous pouvez également ajouter une ou plusieurs règles sortantes si nécessaire.

7. Lorsque vous avez terminé, choisissez Create security group (Créer un groupe de sécurité).

Vous pourrez utiliser ce nouveau groupe de sécurité VPC lorsque vous créerez un autre cluster de bases de données Neptune.

Si vous utilisez un VPC par défaut, un groupe de sous-réseaux par défaut couvrant l'ensemble des sous-réseaux du VPC a déjà été créé pour vous. Lorsque vous choisissez Créer une base de données dans la console Neptune, le VPC par défaut est utilisé, sauf si vous en spécifiez un autre.

Vérifier que le VPC est compatible avec DNS

Le DNS (Domain Name System) est une norme permettant la résolution des noms utilisés sur Internet en leurs adresses IP correspondantes. Un nom d'hôte DNS nomme de façon unique un ordinateur et se compose d'un nom d'hôte et d'un nom de domaine. Les serveurs DNS résolvent les noms d'hôte DNS en adresses IP correspondantes.

Vérifiez que les noms d'hôte DNS et la résolution DNS sont tous les deux activés dans votre VPC. Les attributs de réseau VPC `enableDnsHostnames` et `enableDnsSupport` doivent être définis sur `true`. Pour afficher et modifier ces attributs, accédez à la console VPC à l'adresse https://console.aws.amazon.com/vpc/.

Pour plus d'informations, consultez Utilisation de DNS avec votre VPC.

> **Note**
>
> Si vous utilisez Route 53, vérifiez que votre configuration ne remplace pas les attributs réseau DNS dans votre VPC.

Connexion à un cluster Amazon Neptune

Après avoir créé un cluster Neptune, vous devez configurer l'accès au réseau afin que vos applications puissent atteindre le point de terminaison du cluster. Les sections suivantes décrivent les options de connectivité réseau. Pour plus d'informations sur l'interrogation de vos données après votre connexion, consultezAccès aux données du graphe.

Options de connectivité réseau

Les clusters de bases de données Amazon Neptune peuvent uniquement être créés dans un Amazon Virtual Private Cloud (Amazon VPC). Ses points de terminaison ne sont accessibles qu'au sein de ce VPC, sauf si vous activez et configurez les points de terminaison publics Neptune pour le cluster de base de données.

Vous pouvez configurer l'accès réseau à votre cluster de base de données Neptune dans son VPC de plusieurs manières :

- Connexion à partir d'une instance Amazon EC2 dans le même VPC
- Connexion à partir d'une instance Amazon EC2 dans un autre VPC
- Connexion à partir d'un réseau privé
- Connexion depuis un point de terminaison public

Vérifiez votre connexion

Après avoir configuré l'accès au réseau, vous pouvez vérifier que votre connexion fonctionne en appelant le point de terminaison d'état de l'instance. Une réponse positive confirme que votre client peut accéder au cluster Neptune.

AWS CLI

Exécutez la commande suivante :

```
aws neptunedata get-engine-status \
  --endpoint-url https://your-neptune-endpoint:port \
  --region us-east-1
```

Pour plus d'informations, consultez get-engine-status dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=10, retries={'total_max_attempts': 1})
)

response = client.get_engine_status()

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/status \
  --region us-east-1 \
  --service neptune-db
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultezUtiliser `awscurl` avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

curl

Exécutez la commande suivante :

```
curl -G https://your-neptune-endpoint:port/status
```

> **Note**
>
> Cette commande ne fonctionne que lorsque l'authentification IAM n'est pas activée sur votre cluster. Si l'authentification IAM est activée, utilisez awscurl ou à la AWS CLI place.

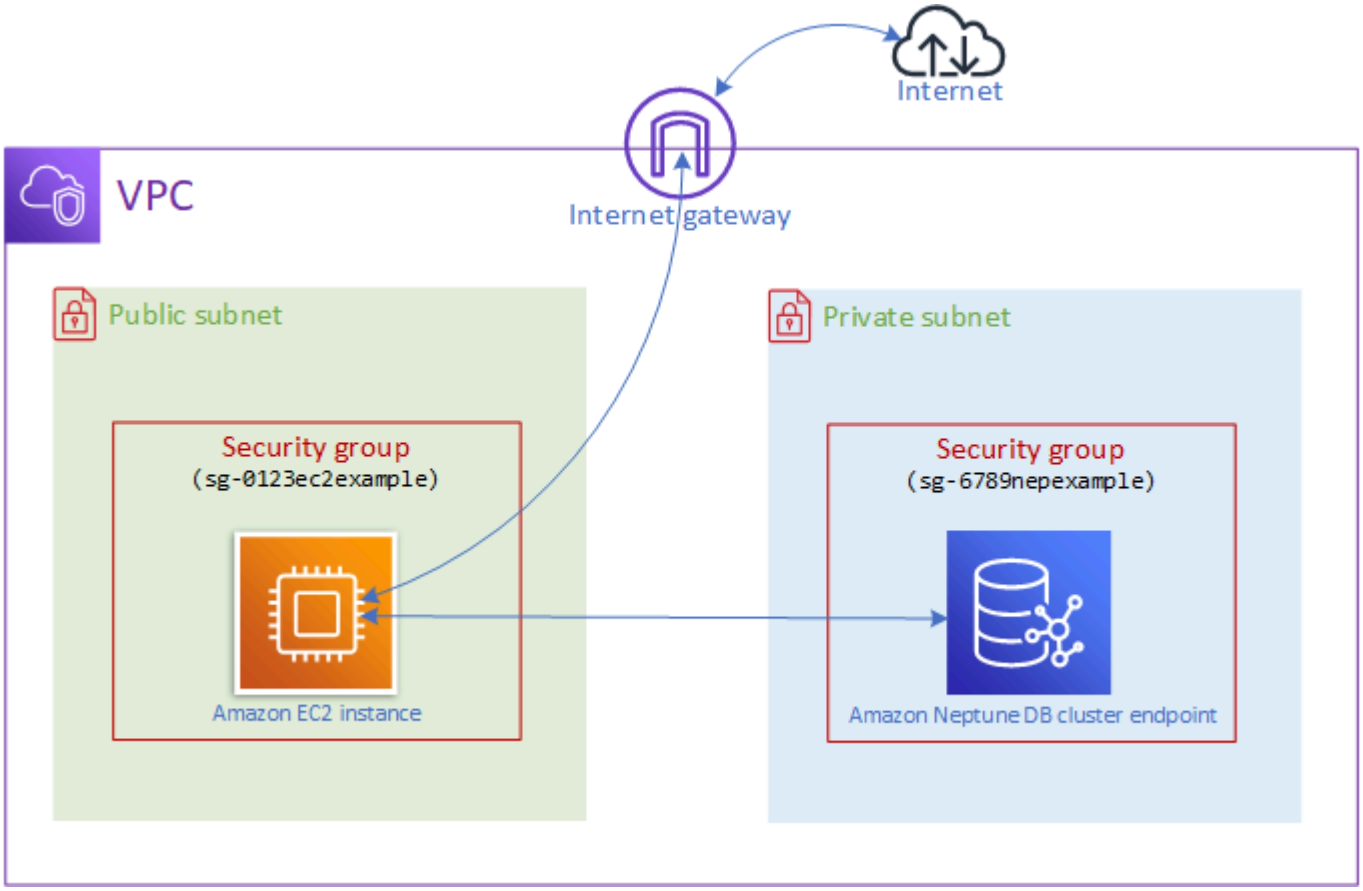
Un cluster sain renvoie une réponse JSON qui inclut`"status": "healthy"`. Par exemple :

```
{
    "status": "healthy",
    "startTime": "Thu Aug 24 21:07:13 UTC 2023",
    "dbEngineVersion": "1.3.1.0.R1",
    "role": "writer",
    "dfeQueryEngine": "viaQueryHint",
    "gremlin": { "version": "tinkerpop-3.7.2" },
    "sparql": { "version": "sparql-1.1" },
    "opencypher": { "version": "Neptune-9.0.20190305-1.0" },
    "labMode": { ... },
    "features": { ... },
    "settings": { ... }
}
```

Si vous ne pouvez pas atteindre le point de terminaison, vérifiez que la configuration de votre réseau autorise le trafic sur le port 8182 (ou le port que vous avez configuré) et que les règles de votre groupe de sécurité autorisent les connexions entrantes en provenance de votre client. Pour de plus amples informations, veuillez consulter Sécurisation de l'accès à Neptune. Pour obtenir de l'aide sur la configuration des SDK AWS CLI, ou`curl`, voirUtilisation des outils de ligne de commande.

Connexion d'une instance Amazon EC2 au cluster Amazon Neptune dans le même VPC

L'un des moyens les plus courants de connexion à une base de données Neptune consiste à utiliser une instance Amazon EC2 située dans le même VPC que le cluster de bases de données Neptune. Par exemple, l'instance EC2 pourrait exécuter un serveur web connecté à Internet. Dans ce cas, seule l'instance EC2 a accès au cluster de bases de données Neptune, et Internet n'a accès qu'à l'instance EC2 :



Pour activer cette configuration, vous devez configurer les groupes de sécurité VPC et les groupes de sous-réseaux appropriés. Le serveur web étant hébergé dans un sous-réseau public, il peut communiquer avec Internet, et l'instance de cluster Neptune est hébergée dans un sous-réseau privé afin d'assurer sa sécurité. Consultez Configureation du VPC Amazon où se trouve le cluster de bases de données Amazon Neptune.

Pour que l'instance Amazon EC2 se connecte à votre point de terminaison Neptune, par exemple sur le port 8182, vous devez configurer un groupe de sécurité à cet effet. Si l'instance Amazon EC2 utilise un groupe de sécurité nommé, par exemple `ec2-sg1`, vous devez créer un autre groupe de sécurité Amazon EC2 (disons `db-sg1`) avec des règles entrantes pour le port `8182` et `ec2-sg1` comme source. Ensuite, ajoutez `db-sg1` au cluster Neptune pour autoriser la connexion.

Après avoir créé l'instance Amazon EC2, vous pouvez vous y connecter via SSH et vous connecter au cluster de bases de données Neptune. Pour plus d'informations sur la connexion à une instance EC2 via SSH, consultez la section Connect to your Linux User Guide d'utilisation d'Amazon EC2.

Si vous utilisez une ligne de commande Linux ou macOS pour vous connecter à l'instance EC2, vous pouvez coller la commande SSH depuis l'élément SSHAccess dans la section Outputs de la pile.

CloudFormation Le fichier PEM doit se trouver dans le répertoire actuel et les autorisations pour ce fichier doivent être définies sur 400 (`chmod 400 keypair.pem`).

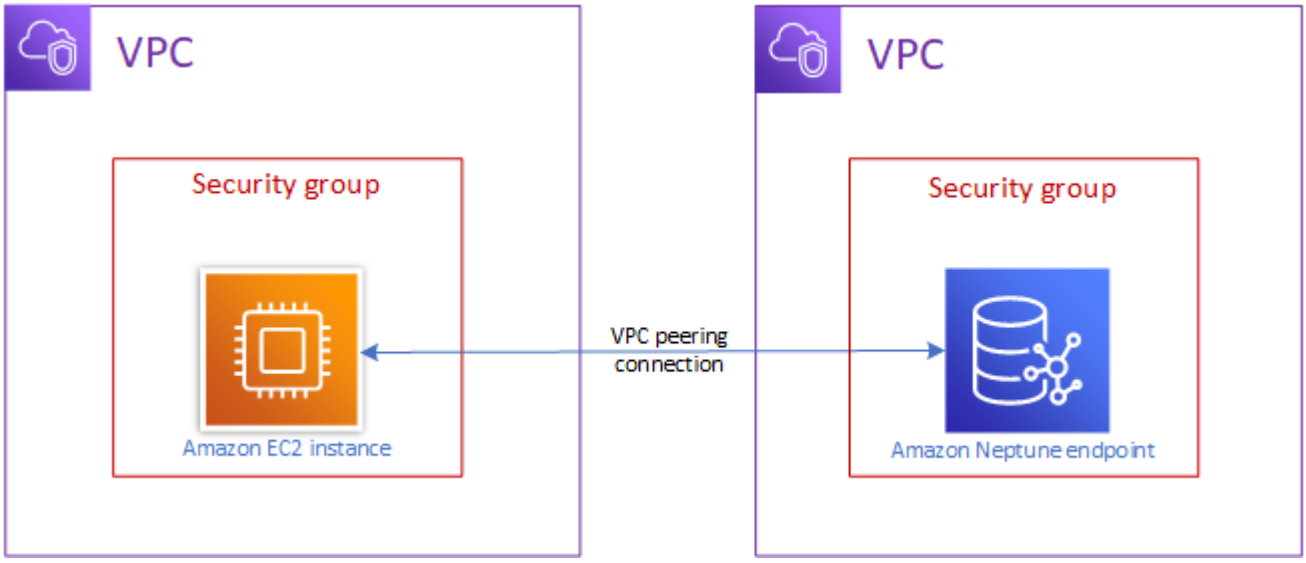
Pour créer un VPC avec des sous-réseaux publics et privés

1. Connectez-vous à la console Amazon VPC AWS Management Console et ouvrez-la à l'adresse. https://console.aws.amazon.com/vpc/
2. Dans le coin supérieur droit du AWS Management Console, choisissez la région dans laquelle créer votre VPC.
3. Dans Tableau de bord VPC, choisissez Lancer l'assistant VPC.
4. Renseignez la zone Paramètres VPC de la page Créer un VPC :
    a. SousRessources à créer, choisissezVPC, sous-réseaux, etc..
    b. Laissez le badge par défaut tel quel, entrez le nom de votre choix, ou décochez la Auto-generatecase pour désactiver la génération de tags nominatifs.
    c. Laissez la valeur du bloc d'adresse CIDR IPv4 : `10.0.0.0/16`.
    d. Laissez la valeur du bloc d'adresse CIDR IPv6 : Aucun bloc d'adresse CIDR IPv6.
    e. Laissez l'option Location : Par défaut.
    f. Dans le champ Nombre de zones de disponibilité, laissez 2.
    g. Laissez la valeur Aucune pour Passerelles NAT ($), sauf si vous avez besoin d'une ou de plusieurs passerelles NAT.
    h. Définissez Points de terminaison VPC sur Aucun, sauf si vous utilisez Amazon S3.
    i. Les cases Activer les noms d'hôte DNS et Activer la résolution DNS doivent toutes deux être cochées.
5. Sélectionnez Create VPC (Créer un VPC).

Connexion d'une instance Amazon EC2 à un cluster Amazon Neptune dans un autre VPC

Un cluster de base de données Amazon Neptune ne peut être créé que dans un Amazon Virtual Private Cloud (Amazon VPC), et ses points de terminaison sont accessibles au sein de ce VPC, généralement à partir d'une instance Amazon Elastic Compute Cloud (Amazon EC2) exécutée dans ce VPC. Il est également possible d'y accéder via un point de terminaison public. Pour plus d'informations sur les points de terminaison publics, consultez la section Points de terminaison publics Neptune.

Lorsque le cluster de bases de données se trouve dans un VPC différent de l'instance EC2 que vous utilisez pour y accéder, vous pouvez utiliser l'appairage de VPC pour établir la connexion :



Une connexion d'appairage de VPC est une connexion de mise en réseau entre deux VPC qui achemine le trafic de l'un à l'autre de manière privée, de sorte que les instances de l'un ou l'autre de ces VPC communiquent comme si elles se trouvaient dans le même réseau. Vous pouvez créer une connexion d'appairage VPC entre des VPC de votre compte, entre un VPC de votre compte AWS et un VPC d'un autre compte AWS , ou avec un VPC d'une autre région. AWS

AWS utilise l'infrastructure existante d'un VPC pour créer une connexion d'appairage VPC. Il ne s'agit ni d'une passerelle ni d'une connexion AWS Site-to-Site VPN, et elle ne repose pas sur un matériel physique distinct. Il n'y a pas de point unique de défaillance pour la communication ni de goulet d'étranglement en termes de bande passante.

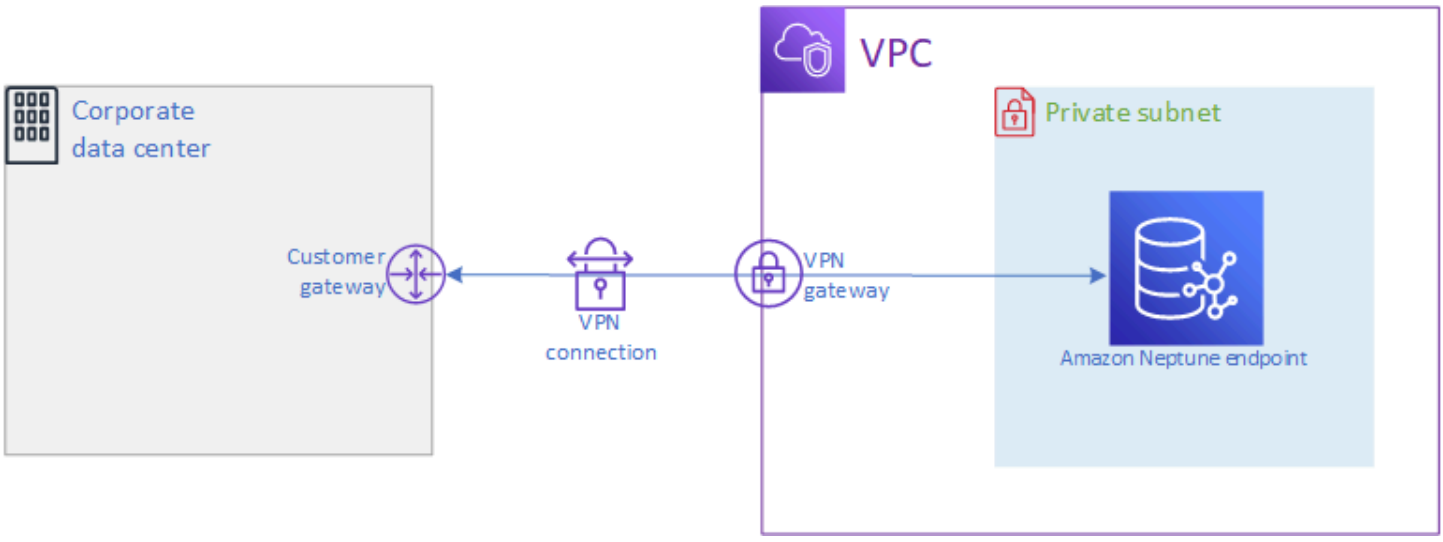
Pour plus d'informations sur les connexions d'appairage de VPC, consultez le Guide sur l'appairage de VPC Amazon.

Connexion à un cluster Amazon Neptune via un réseau privé

Vous pouvez accéder à un cluster de bases de données Neptune à partir d'un réseau privé de deux manières différentes :

- À l'aide d'une connexion AWS Site-to-Site VPN.
- Avec une connexion AWS Direct Connect

Les liens ci-dessus contiennent des informations sur ces méthodes de connexion et sur la façon de les configurer. La configuration d'une AWS Site-to-Site connexion peut ressembler à ceci :



Points de terminaison publics Neptune

Présentation de

Les clusters Amazon Neptune sont généralement déployés au sein de votre VPC et ne sont accessibles qu'à partir de ce VPC. Cela nécessite de configurer des applications et des environnements de développement au sein du VPC ou d'utiliser des services proxy pour se connecter au VPC, ce qui augmente le temps et les coûts de configuration.

Les points de terminaison publics simplifient cette expérience en permettant des connexions directes à Neptune via Internet, ce qui permet de démarrer plus facilement avec des bases de données de graphes sans connaissances spécifiques en matière de réseau.

Quand utiliser des points de terminaison publics

Envisagez d'utiliser des points de terminaison publics dans les scénarios suivants :

- Vous souhaitez tester rapidement Neptune dans un environnement de développement ou de test sans configuration réseau complexe
- Vous n'avez pas de connaissances spécialisées AWS en matière de réseaux
- Le niveau de sécurité de votre application ne nécessite pas de VPC privé
- Vous devez vous connecter à Neptune depuis votre environnement de développement local

Considérations sur la sécurité

Lorsque vous utilisez des points de terminaison publics, tenez compte des considérations de sécurité suivantes :

- L'authentification IAM est requise pour les clusters dont les points de terminaison publics sont activés.
- L'accès à la base de données est contrôlé par le groupe de sécurité qu'elle utilise.
- Vous pouvez restreindre les adresses IP autorisées à se connecter à votre cluster.
- Vous pouvez utiliser les politiques IAM pour contrôler qui peut créer ou modifier des clusters avec un accès public. Voir : Restreindre la création d'accès public

Activation des points de terminaison publics

Par défaut, les nouvelles bases de données Neptune sont créées avec les points de terminaison publics désactivés. Vous devez explicitement activer l'accès public lors de la création ou de la modification d'un cluster.

Les points de terminaison publics sont pris en charge à partir de la version 1.4.6.x du moteur Neptune. Vous devez mettre à niveau les clusters existants vers au moins cette version pour utiliser cette fonctionnalité.

Le paramètre de point de terminaison public est disponible sur l'instance Neptune et non sur le cluster Neptune. Par conséquent, un cluster Neptune peut exister avec certaines instances avec des points de terminaison publics et d'autres non. Toutefois, nous ne recommandons pas d'utiliser un tel paramètre. Pour plus d'informations à ce sujet, reportez-vous à : Fonctionnement des points de terminaison publics

Conditions préalables

Paramètre d'authentification IAM sur le cluster Neptune

Avant d'activer des points de terminaison publics sur une instance Neptune, assurez-vous que votre cluster prend en charge l'authentification IAM. Si ce n'est pas le cas, activez-le à l'aide de la commande suivante :

```
aws neptune modify-db-cluster \
  --region us-west-2 \
  --engine graphdb \
```

```
  --engine-version 1.4.6.x \
  --db-cluster-identifier neptune-public-endpoint \
  --enable-iam-database-authentication
```

Paramètres réseau

1. Assurez-vous que votre VPC possède des sous-réseaux qui permettent le routage public (il comporte une entrée pour une passerelle Internet dans la table de routage des sous-réseaux). Si vous ne fournissez aucun `db-subnet-group-name` paramètre lors de la création du cluster, le groupe de sous-réseaux par défaut est sélectionné pour la création du cluster.
2. Assurez-vous que le groupe de sécurité attaché au cluster autorise le trafic entrant pour les plages d'adresses IP autorisées et les ports autorisés. Par exemple, si vous souhaitez autoriser le trafic TCP provenant de toutes les adresses IP à se connecter à l'instance Neptune exécutée sur le port 8182, la règle entrante doit être la suivante :
   a. Type : Tout TCP
   b. Protocole : TCP
   c. Portée de ports : 8182
   d. Bloc CIDR : 0,0.0. 0/0

> **Note**
>
> Bien que vous puissiez définir la plage de blocs CIDR sur 0,0.0. 0/0, nous vous recommandons de le réduire à une plage d'adresses IP spécifique de votre application cliente pour une meilleure sécurité.

Création d'une nouvelle instance avec des points de terminaison publics

Vous pouvez créer une nouvelle instance Neptune avec des points de terminaison publics à l'aide de la console de AWS gestion, de la AWS CLI ou du SDK. AWS

À l'aide de la AWS CLI :

```
aws neptune create-db-instance \
  --region us-west-2 \
  --engine graphdb \
  --engine-version 1.4.6.x \
  --db-cluster-identifier neptune-public-endpoint \
```

```
  --publicly-accessible
```

Modification d'une instance existante pour un accès public

Pour modifier une instance Neptune existante afin de permettre l'accès public :

```
aws neptune modify-db-instance \
  --region us-west-2 \
  --engine graphdb \
  --engine-version 1.4.6.x \
  --db-instance-identifier neptune-public-endpoint \
  --publicly-accessible
```

> **Note**
>
> L'accès public est activé au niveau de l'instance, et non au niveau du cluster. Pour garantir que votre cluster est toujours accessible via des points de terminaison publics, l'accès public doit être activé pour toutes les instances du cluster.

Utilisation des points de terminaison publics

Pour vérifier si votre base de données est accessible, vérifiez son état à l'aide de l'`NeptuneData`API AWS CLI :

```
aws neptunedata get-engine-status \
  --endpoint-url https://my-cluster-name.cluster-abcdefgh1234.us-
east-1.neptune.amazonaws.com:8182
```

Si la base de données est accessible, la réponse est la suivante :

```
{
    "status": "healthy",
    "startTime": "Sun Aug 10 06:54:15 UTC 2025",
    "dbEngineVersion": "1.4.6.0.R1",
    "role": "writer",
    "dfeQueryEngine": "viaQueryHint",
    "gremlin": {
        "version": "tinkerpop-3.7.1"
    },
```

```
    "sparql": {
        "version": "sparql-1.1"
    },
    "opencypher": {
        "version": "Neptune-9.0.20190305-1.0"
    },
    "labMode": {
        "ObjectIndex": "disabled",
        "ReadWriteConflictDetection": "enabled"
    },
    "features": {
        "SlowQueryLogs": "disabled",
        "InlineServerGeneratedEdgeId": "disabled",
        "ResultCache": {
            "status": "disabled"
        },
        "IAMAuthentication": "disabled",
        "Streams": "disabled",
        "AuditLog": "disabled"
    },
    "settings": {
        "StrictTimeoutValidation": "true",
        "clusterQueryTimeoutInMs": "120000",
        "SlowQueryLogsThreshold": "5000"
    }
}
```

Exemples d'interrogation de la base de données

AWS INTERFACE DE LIGNE DE COMMANDE (CLI)

```
aws neptunedata execute-open-cypher-query \
--open-cypher-query "MATCH (n) RETURN n LIMIT 10" \
--endpoint-url https://my-cluster-name.cluster-abcdefgh1234.us-
east-1.neptune.amazonaws.com:8182
```

Python

```
import boto3
import json
from botocore.config import Config

# Configuration - Replace with your actual Neptune cluster details
```

```
cluster_endpoint = "my-cluster-name.cluster-abcdefgh1234.my-
region.neptune.amazonaws.com"
port = 8182
region = "my-region"

# Configure Neptune client
# This disables retries and sets the client timeout to infinite
#     (relying on Neptune's query timeout)
endpoint_url = f"https://{cluster_endpoint}:{port}"
config = Config(
    region_name=region,
    retries={'max_attempts': 1},
    read_timeout=None
)

client = boto3.client("neptunedata", config=config, endpoint_url=endpoint_url)

cypher_query = "MATCH (n) RETURN n LIMIT 5"
try:
    response = client.execute_open_cypher_query(openCypherQuery=cypher_query)
    print("openCypher Results:")
    for item in response.get('results', []):
        print(f"  {item}")
except Exception as e:
    print(f"openCypher query failed: {e}")
```

JavaScript

```
import {
    NeptunedataClient,
    GetPropertygraphSummaryCommand
} from "@aws-sdk/client-neptunedata";
import { inspect } from "util";
import { NodeHttpHandler } from "@smithy/node-http-handler";

/**
 * Main execution function
 */
async function main() {
    // Configuration - Replace with your actual Neptune cluster details
    const clusterEndpoint = 'my-cluster-name.cluster-abcdefgh1234.my-
region.neptune.amazonaws.com';
    const port = 8182;
```

```
    const region = 'my-region';

    // Configure Neptune client
    // This disables retries and sets the client timeout to infinite
    //     (relying on Neptune's query timeout)
    const endpoint = `https://${clusterEndpoint}:${port}`;
    const clientConfig = {
        endpoint: endpoint,
        sslEnabled: true,
        region: region,
        maxAttempts: 1,  // do not retry
        requestHandler: new NodeHttpHandler({
            requestTimeout: 0  // no client timeout
        })
    };

    const client = new NeptunedataClient(clientConfig);
    try {
        try {
            const command = new GetPropertygraphSummaryCommand({ mode: "basic" });
            const response = await client.send(command);
            console.log("Graph Summary:", inspect(response.payload, { depth: null }));
        } catch (error) {
            console.log("Property graph summary failed:", error.message);
        }
    } catch (error) {
        console.error("Error in main execution:", error);
    }
}

// Run the main function
main().catch(console.error);
```

Go

```
package main
import (
    "context"
    "fmt"
    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/neptunedata"
    "os"
```

```
    "encoding/json"
    "net/http"
)

func main() {
    // Configuration - Replace with your actual Neptune cluster details
    clusterEndpoint := "my-cluster-name.cluster-abcdefgh1234.my-
region.neptune.amazonaws.com"
    port := 8182
    region := "my-region"

    // Configure Neptune client
    // Configure HTTP client with no timeout
    //    (relying on Neptune's query timeout)
    endpoint := fmt.Sprintf("https://%s:%d", clusterEndpoint, port)
    // Load AWS SDK configuration
    sdkConfig, _ := config.LoadDefaultConfig(
        context.TODO(),
        config.WithRegion(region),
        config.WithHTTPClient(&http.Client{Timeout: 0}),
    )

    // Create Neptune client with custom endpoint
    client := neptunedata.NewFromConfig(sdkConfig, func(o *neptunedata.Options) {
        o.BaseEndpoint = aws.String(endpoint)
        o.Retryer = aws.NopRetryer{} // Do not retry calls if they fail
    })

    gremlinQuery :=
 "g.addV('person').property('name','charlie').property(id,'charlie-1')"
    serializer := "application/vnd.gremlin-v1.0+json;types=false"

    gremlinInput := &neptunedata.ExecuteGremlinQueryInput{
        GremlinQuery: &gremlinQuery,
        Serializer:   &serializer,
    }
    gremlinResult, err := client.ExecuteGremlinQuery(context.TODO(), gremlinInput)
    if err != nil {
        fmt.Printf("Gremlin query failed: %v\n", err)
    } else {
        var resultMap map[string]interface{}
        err = gremlinResult.Result.UnmarshalSmithyDocument(&resultMap)
        if err != nil {
            fmt.Printf("Error unmarshaling Gremlin result: %v\n", err)
```

```
        } else {
            resultJSON, _ := json.MarshalIndent(resultMap, "", "  ")
            fmt.Printf("Gremlin Result: %s\n", string(resultJSON))
        }
    }
}
```

Comment fonctionnent les points de terminaison publics

Lorsqu'une instance Neptune est accessible au public :

- Son point de terminaison DNS est résolu vers l'adresse IP privée depuis le VPC du cluster de base de données.
- Il est résolu vers l'adresse IP publique depuis l'extérieur du VPC du cluster.
- L'accès est contrôlé par le groupe de sécurité attribué au cluster.
- Seules les instances accessibles au public sont accessibles via Internet.

Comportement du terminal du lecteur

- Si toutes les instances du lecteur sont accessibles au public, le point de terminaison du lecteur sera toujours résolu via l'Internet public.
- Si seules certaines instances de lecteur sont accessibles au public, le point de terminaison du lecteur ne sera résolu publiquement que s'il sélectionne une instance accessible au public pour répondre à la requête de lecture.

Comportement des terminaux du cluster

- Le point de terminaison du cluster de base de données est toujours le point de terminaison de l'instance du rédacteur.
- Si le point de terminaison public est activé sur l'instance du rédacteur, le point de terminaison du cluster sera accessible au public, sinon il ne le sera pas.

Comportement après le basculement d'un cluster

- Un cluster Neptune peut avoir des instances dans différents paramètres accessibles au public.

- Si un cluster possède un rédacteur public et un lecteur non public, publiez un basculement du cluster, le nouveau rédacteur (lecteur précédent) devient non public et le nouveau lecteur (rédacteur précédent) devient public.

Exigences en matière de configuration du réseau

Pour que les points de terminaison publics fonctionnent correctement :

1. Les instances Neptune doivent se trouver dans des sous-réseaux publics au sein de votre VPC.
2. Les tables de routage associées à ces sous-réseaux doivent disposer d'une route vers une passerelle Internet pour la version 0.0.0. 0/0.
3. Le groupe de sécurité doit autoriser l'accès à partir des adresses IP publiques ou des plages d'adresses CIDR auxquelles vous souhaitez accorder l'accès.

Restreindre la création d'accès public

Vous pouvez utiliser les politiques IAM pour restreindre les personnes autorisées à créer ou à modifier des clusters Neptune avec un accès public. L'exemple de politique suivant refuse la création d'instances Neptune accessibles au public :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "rds:CreateDBInstance",
        "rds:ModifyDBInstance",
        "rds:RestoreDBInstanceFromDBSnapshot",
        "rds:RestoreDBInstanceToPointInTime"
      ],
      "Resource": "*",
      "Condition": {
        "Bool": {
          "rds:PubliclyAccessible": true
        }
      }
```

```
    }
  ]
}
```

En savoir plus sur la clé de condition `rds:PublicAccessEnabled` IAM : Amazon RDS Service Authorization Reference

CloudFormation Prise en charge de par la

Vous pouvez l'utiliser CloudFormation pour lancer des clusters Neptune avec des points de terminaison publics activés en spécifiant le `PubliclyAccessible` paramètre dans votre modèle. CloudFormation

Compatibilité avec les fonctionnalités de Neptune

Un cluster dont les points de terminaison publics sont activés prend en charge toutes les fonctionnalités de Neptune prises en charge par VPC-only un cluster, notamment :

- Établi Neptune
- Intégration de la recherche en texte intégral
- Neptune Streams
- Points de terminaison personnalisés
- Neptune sans serveur
- Explorateur de graphes

Tarification

Les terminaux publics sont disponibles sans frais supplémentaires au-delà de la tarification standard de Neptune. Cependant, la connexion de votre environnement local à Neptune via une adresse IP publique peut entraîner une augmentation des coûts de transfert de données.

Sécurisation de l'accès à un cluster Amazon Neptune

Vous pouvez sécuriser vos clusters Amazon Neptune de plusieurs façons.

Utilisation de politiques IAM pour restreindre l'accès à un cluster de bases de données Neptune

Pour contrôler qui peut effectuer les actions de gestion de Neptune sur les clusters de bases de données et les instances de base de données Neptune, utilisez Gestion des identités et des accès AWS (IAM).

Lorsque vous utilisez un compte IAM pour accéder à la console Neptune, vous devez d'abord vous connecter à l'aide de votre compte IAM avant d'ouvrir AWS Management Console la console Neptune à l'adresse. https://console.aws.amazon.com/neptune/home

Lorsque vous vous connectez à AWS l'aide d'informations d'identification IAM, votre compte IAM doit disposer de politiques IAM qui accordent les autorisations requises pour effectuer les opérations de gestion de Neptune. Pour de plus amples informations, veuillez consulter Utilisation de différents types de politique IAM pour contrôler l'accès à Neptune.

Utilisation de groupes de sécurité VPC pour restreindre l'accès à un cluster de bases de données Neptune

Les clusters de bases de données Neptune doivent être créés dans un Amazon Virtual Private Cloud (VPC). Pour contrôler les appareils et les instances EC2 qui peuvent ouvrir des connexions au point de terminaison et au port de l'instance de base de données pour les clusters de bases de données Neptune dans un VPC, utilisez un groupe de sécurité VPC. Pour plus d'informations sur le VPCs, consultez Créez un groupe de sécurité à l'aide de la console VPC..

> **Note**
>
> Pour vous connecter à votre cluster Neptune, vous devez exposer le port de base de données du cluster (8182 par défaut) pour les règles entrantes et sortantes afin de permettre une connectivité adéquate.

Utilisation de l'authentification IAM pour restreindre l'accès à un cluster de bases de données Neptune

Si vous activez l'authentification Gestion des identités et des accès AWS (IAM) dans un cluster de base de données Neptune, toute personne accédant au cluster de base de données doit d'abord

être authentifiée. Pour plus d'informations sur la configuration de l'authentification IAM, consultez Authentification de votre base de données Amazon Neptune avec Gestion des identités et des accès AWS.

Pour plus d'informations sur l'utilisation d'informations d'identification temporaires pour l'authentification, y compris des exemples pour le AWS CLI AWS Lambda, et Amazon EC2, consultez. the section called "Informations d'identification temporaires"

Les liens suivants fournissent des informations supplémentaires sur la connexion à Neptune à l'aide de l'authentification IAM avec les différents langages de requête :

Utilisation de Gremlin avec l'authentification IAM

- the section called "Connexion à partir de la console Garmlin"
- the section called "Connexion à l'aide de Gremlin Java"
- the section called "Python Garmlin"
- the section called "Javascript"
- the section called "Garmlin Go"
- the section called "Garmlin.net"

Utilisation d'openCypher avec l'authentification IAM

- the section called "Exemple Java Bolt avec authentification IAM"
- the section called "Exemple Python Bolt avec authentification IAM"
- the section called "Node.js avec authentification IAM"
- the section called "Exemple .NET Bolt avec authentification IAM"
- the section called "Exemple de Bolt d'authentification Golang IAM"

Utilisation de SPARQL avec l'authentification IAM

- the section called "Connexion avec SPARQL Java"
- the section called "Connexion avec SPARQL et Node.js"
- the section called "Connexion avec Python"

Accès aux données graphiques dans Amazon Neptune

Vous pouvez interagir avec un cluster de base de données Amazon Neptune après avoir établi la connectivité réseau. Si vous n'avez pas encore configuré l'accès à votre cluster, consultezConnexion à un cluster. Les sections suivantes décrivent les outils et les langages de requête que vous pouvez utiliser pour charger des données, exécuter des requêtes et gérer votre graphe.

Outils pour accéder à Neptune

Neptune prend en charge plusieurs outils pour soumettre des requêtes et gérer les données de vos graphes :

- AWS CLI— Utilisez les `aws neptunedata` commandes pour exécuter des requêtes Gkremlin et OpenCypher, vérifier l'état du moteur, gérer les chargements groupés, etc. Pour plus d'informations, consultez neptunedatale manuel de référence des AWS CLI commandes.
- AWS SDK — Utilisez l'API Neptune Data via les SDK pour exécuter AWS des requêtes par programmation. Des SDK sont disponibles pour G705 et OpenCypher.
- **`curl`**et **`awscurl`** — `curl` À utiliser pour envoyer des requêtes HTTP directement aux points de terminaison Neptune. Si l'authentification IAM est activée, utilisez awscurl ou `curl` 7.75.0+ avec la possibilité de signer les demandes. `--aws-sigv4` Pour de plus amples informations, veuillez consulter Utiliser `awscurl` avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.
- Carnets Neptune : utilisez les blocs-notes Neptune pour exécuter des requêtes interactives dans un environnement Jupyter avec des visualisations intégrées. Pour de plus amples informations, veuillez consulter Utilisation de Neptune avec des carnets graphiques.
- Pilotes : connectez-vous à l'aide de pilotes spécifiques à chaque langue de requête. Pour G705, utilisez TinkerPop-compliant les pilotes disponibles dans le TinkerPop projet Apache. Pour OpenCypher, utilisez les pilotes du protocole Bolt comme décrit dans. Utilisation du protocole Bolt Pour SPARQL, utilisez des outils compatibles tels que RDF4J pour Java, comme décrit dans. Java

Utilisation d'un langage de requête pour accéder aux données de graphe dans le cluster de bases de données Neptune

Une fois connecté, vous pouvez utiliser les langages de requête Gremlin et openCypher pour créer et interroger un graphe de propriétés, ou le langage de requête SPARQL pour créer et interroger un graphe contenant des données RDF.

Langages de requête orientés graphe pris en charge par Neptune

- Gremlin est un langage de parcours de graphe pour les graphes de propriétés. Dans Gremlin, une requête est une traversée composée d'étapes distinctes, chacune suivant une arête jusqu'à un nœud. Pour plus d'informations, consultez la documentation de Gremlin sur Apache TinkerPop.

  L'implémentation Neptune de G705 présente certaines différences par rapport aux autres implémentations, en particulier lorsque vous l'utilisez Gremlin-Groovy (requêtes G705 envoyées sous forme de texte sérialisé). Pour de plus amples informations, veuillez consulter Conformité d'Amazon Neptune avec les normes Gremlin.

  Consultez Utilisation de Gremlin pour démarrer.

- openCypher est un langage de requête déclaratif pour les graphes de propriétés initialement développé par Neo4j, puis rendu open source en 2015. Il a contribué au projet openCypher sous une licence open source Apache 2. Sa syntaxe est documentée dans Cypher Query Language Reference, Version 9 (Référence du langage de requête Cypher, version 9).

  Consultez Utilisation d'openCypher pour démarrer.

- Le SPARQL est un langage de requête déclaratif pour les données RDF, basé sur la correspondance de modèles de graphes normalisée par le World Wide Web Consortium (W3C) et décrite dans la présentation de SPARQL 1.1 et dans la spécification du langage de requête SPARQL 1.1.

  Consultez Utilisation de SPARQL pour démarrer.

> **Note**
>
> Vous pouvez accéder aux données du graphe de propriétés dans Neptune à la fois à l'aide de Gremlin et d'openCypher, mais pas avec SPARQL. De même, vous ne pouvez accéder aux données RDF qu'à l'aide de SPARQL, et non de Gremlin ou d'openCypher.

Utilisation d'outils de ligne de commande pour accéder à Amazon Neptune

Vous pouvez utiliser les AWS CLI AWS SDK et les outils HTTP tels que `curl` et `awscurl` pour envoyer des requêtes à votre cluster de base de données Neptune. Les sections suivantes montrent comment configurer chaque outil et exécuter des requêtes Gkremlin et OpenCypher de base.

Utilisation de AWS CLI

Les `aws neptunedata` commandes vous permettent d'exécuter des requêtes G705 et OpenCypher, de vérifier l'état du moteur, de gérer les chargements groupés, etc. Pour la référence complète des commandes, voir neptunedatala référence des AWS CLI commandes.

Les exemples suivants montrent comment exécuter une requête de base :

Gremlin

```
aws neptunedata execute-gremlin-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --region us-east-1 \
  --gremlin-query "g.V().limit(1)"
```

Pour plus d'informations, consultez execute-gremlin-query dans la référence des commandes. AWS CLI

openCypher

```
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --region us-east-1 \
  --open-cypher-query "MATCH (n) RETURN n LIMIT 1"
```

Pour plus d'informations, voir execute-open-cypher-query dans la référence des commandes. AWS CLI

Utilisation de AWS Kit SDK

Vous pouvez utiliser l'API Neptune Data via les AWS SDK pour exécuter des requêtes par programmation. Les exemples Python suivants montrent comment exécuter une requête de base :

Gremlin

```
import boto3
import json
from botocore.config import Config

# Disable the client-side read timeout and retries so that
# Neptune's server-side neptune_query_timeout controls query duration.
```

```
client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_gremlin_query(
    gremlinQuery='g.V().limit(1)',
    serializer='application/vnd.gremlin-v3.0+json;types=false'
)

print(json.dumps(response['result'], indent=2))
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

openCypher

```
import boto3
import json
from botocore.config import Config

# Disable the client-side read timeout and retries so that
# Neptune's server-side neptune_query_timeout controls query duration.
client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_query(
    openCypherQuery='MATCH (n) RETURN n LIMIT 1'
)

print(json.dumps(response['results'], indent=2))
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

Utilisation de curl et awscurl

L'outil de ligne de commande curl envoie des requêtes HTTP directement aux points de terminaison Neptune. Si l'authentification IAM est activée, utilisez awscurl ou `curl` 7.75.0+ avec la possibilité de signer les demandes. `--aws-sigv4` Pour de plus amples informations, veuillez consulter Utiliser

awscurl avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

Configuration de **curl** pour HTTPS

Pour se connecter via HTTPS (comme Neptune l'exige dans la plupart des régions), il curl faut avoir accès aux certificats appropriés. Pour plus d'informations sur la façon d'obtenir des certificats et de les formater dans un magasin d'autorité de certification (CA), consultez la section Vérification des certificats SSL dans la curl documentation.

Vous pouvez spécifier l'emplacement de ce magasin de certificats CA à l'aide de la variable d'CURL_CA_BUNDLEenvironnement. Sous Windows, curl le recherche automatiquement dans un fichier nommé curl-ca-bundle.crt. Il examine d'abord dans le même répertoire que curl.exe, puis ailleurs sur le chemin. Pour plus d'informations, consultez SSL Certificate Verification (Vérification des certificats SSL).

Dans la mesure où curl peut localiser les certificats appropriés, il gère les connexions HTTPS comme des connexions HTTP, sans paramètres supplémentaires. Les exemples de cette documentation sont basés sur ce scénario.

Pour plus d'informations sur cet outil, consultez la page de manuel curl et le livre Everything curl.

Exemples de requêtes

Les exemples suivants montrent comment exécuter une requête de base à l'aide de curl et awscurl :

Gremlin (awscurl)

```
awscurl https://your-neptune-endpoint:port/gremlin \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d '{"gremlin":"g.V().limit(1)"}'
```

Gremlin (curl)

```
curl -X POST \
  -d '{"gremlin":"g.V().limit(1)"}' \
  https://your-neptune-endpoint:port/gremlin
```

> **Note**
>
> Plain ne `curl` fonctionne que lorsque l'authentification IAM n'est pas activée sur votre cluster.

openCypher (awscurl)

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "query=MATCH (n) RETURN n LIMIT 1"
```

openCypher (curl)

```
curl -X POST \
  -d "query=MATCH (n) RETURN n LIMIT 1" \
  https://your-neptune-endpoint:port/openCypher
```

> **Note**
>
> Plain ne `curl` fonctionne que lorsque l'authentification IAM n'est pas activée sur votre cluster.

Pour d'autres exemples HTTP de Gkremlin, voirHTTPS REST. Pour d'autres exemples HTTP d'OpenCypher, consultez. Point de terminaison HTTPS

Utilisation de Gkremlin pour accéder aux données graphiques dans Amazon Neptune

Vous pouvez utiliser la console G705 pour tester TinkerPop des graphiques et des requêtes dans un environnement REPL (read-eval-print loop).

Le didacticiel suivant explique comment utiliser la console G705 pour ajouter des sommets, des arêtes, des propriétés, etc. à un graphe Neptune. Il met en évidence certaines différences dans l'implémentation de G705. Neptune-specific

> **Note**
>
> Cet exemple suppose que vous respectez les prérequis suivants :
>
> - Vous vous êtes connecté via SSH à une instance Amazon EC2.
> - Vous avez créé un cluster Neptune comme décrit dans Créer un cluster Neptune.
> - Vous avez installé la console Gremlin, comme décrit dans Installation de la console Gremlin.

Utilisation de la console Gremlin

1. Modifiez les répertoires dans le dossier où les fichiers de la console Gremlin sont décompressés.

```
cd apache-tinkerpop-gremlin-console-3.7.2
```

2. Saisissez la commande suivante pour exécuter la console Gremlin.

```
bin/gremlin.sh
```

   Vous devriez voir la sortie suivante :

```
         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin>
```

   Vous êtes maintenant à l'invite `gremlin>`. Vous entrez les étapes restantes à cette invite.

3. À l'invite de commande `gremlin>`, saisissez le texte suivant pour vous connecter à l'instance de base de données Neptune.

```
:remote connect tinkerpop.server conf/neptune-remote.yaml
```

4. À l'invite `gremlin>`, entrez ce qui suit pour passer en mode distant. Toutes les requêtes Gremlin sont alors envoyées à la connexion distante.

```
:remote console
```

5. Ajoutez un sommet avec une étiquette et une propriété.

```
g.addV('person').property('name', 'justin')
```

Un ID `string` contenant un GUID est affecté au sommet. Tous les ID de sommet sont des chaînes dans Neptune.

6. Ajoutez un sommet avec ID personnalisé.

```
g.addV('person').property(id, '1').property('name', 'martin')
```

La propriété `id` n'est pas indiquée entre guillemets. Il s'agit d'un mot-clé pour l'ID du sommet. Ici, l'ID du sommet est une chaîne contenant le nombre `1`.

Les noms de propriété normaux doivent être indiqués entre guillemets.

7. Modifiez une propriété ou ajoutez une propriété si elle n'existe pas.

```
g.V('1').property(single, 'name', 'marko')
```

Ici, vous modifiez la propriété `name` du sommet de l'étape précédente. Cela supprime toutes les valeurs existantes de la propriété `name`.

Si vous n'aviez pas spécifié `single`, la commande ajouterait plutôt la valeur à la propriété `name` si elle ne la possédait pas déjà.

8. Ajouter une propriété, mais ajouter la valeur de cette propriété si elle possède déjà une valeur

```
g.V('1').property('age', 29)
```

Neptune utilise la cardinalité définie comme action par défaut.

Cette commande ajoute la propriété `age` avec la valeur `29`, mais ne remplace aucune des valeurs existantes.

Si la propriété `age` avait déjà une valeur, cette commande ajouterait `29` à la propriété. Par exemple, si la propriété `age` était `27`, la nouvelle valeur serait `[ 27, 29 ]`.

9. Ajoutez plusieurs sommets.

```
g.addV('person').property(id, '2').property('name', 'vadas').property('age',
 27).iterate()
g.addV('software').property(id, '3').property('name', 'lop').property('lang',
 'java').iterate()
g.addV('person').property(id, '4').property('name', 'josh').property('age',
 32).iterate()
g.addV('software').property(id, '5').property('name', 'ripple').property('lang',
 'java').iterate()
g.addV('person').property(id, '6').property('name', 'peter').property('age', 35)
```

Vous pouvez envoyer plusieurs instructions simultanément à Neptune.

Les instructions peuvent être séparés par un caractère de saut de ligne (`'\n'`), un espace (`' '`), un point-virgule (`'; '`) ou rien (par exemple, `g.addV('person').iterate()g.V()` est valide).

> **Note**
>
> La console Gremlin envoie une commande distincte à chaque caractère de saut de ligne (`'\n'`). Par conséquent, il s'agira d'opérations distinctes dans ce cas. Dans cet exemple, toutes les commandes sont sur des lignes distinctes pour faciliter la lecture. Supprimez les caractères de saut de ligne (`'\n'`) pour l'envoyer comme une commande unique via la console Gremlin.

Toutes les instructions autres que la dernière doivent se terminer par une étape de fin, comme `.next()` ou `.iterate()`, sinon, elles ne s'exécuteront pas. La console Gremlin ne nécessite pas ces étapes de fin. Utilisez `.iterate` chaque fois que vous n'avez pas besoin de sérialiser les résultats.

Toutes les instructions qui sont envoyées ensemble sont incluses dans une unique transaction, et aboutissent ou échouent ensemble.

10. Ajoutez des arêtes.

```
g.V('1').addE('knows').to(__.V('2')).property('weight', 0.5).iterate()
g.addE('knows').from(__.V('1')).to(__.V('4')).property('weight', 1.0)
```

Il existe deux manières différentes d'ajouter un arc.

11. Ajoutez le reste du graphe Modern.

```
g.V('1').addE('created').to(__.V('3')).property('weight', 0.4).iterate()
g.V('4').addE('created').to(__.V('5')).property('weight', 1.0).iterate()
g.V('4').addE('knows').to(__.V('3')).property('weight', 0.4).iterate()
g.V('6').addE('created').to(__.V('3')).property('weight', 0.2)
```

12. Supprimez un sommet.

```
g.V().has('name', 'justin').drop()
```

Supprime le sommet dont la propriété `name` a la valeur `justin`.

> **Important**
>
> Arrêtez-vous ici, et vous avez le graphe complet d'Apache TinkerPop Modern. Les exemples présentés dans la section Traversal de la TinkerPop documentation utilisent le graphe moderne.

13. Exécutez un parcours.

```
g.V().hasLabel('person')
```

Renvoie tous les sommets `person`.

14. Exécutez un parcours avec des valeurs (valueMap()).

```
g.V().has('name', 'marko').out('knows').valueMap()
```

Renvoie les paires clé-valeur de tous les sommets que `marko` « connaît ».

15. Spécifiez plusieurs étiquettes.

```
g.addV("Label1::Label2::Label3")
```

Neptune prend en charge plusieurs étiquettes pour un sommet. Lorsque vous créez une étiquette, vous pouvez spécifier plusieurs étiquettes en les séparant par `::`.

Cet exemple ajoute un sommet avec trois étiquettes différentes.

L'étape `hasLabel` associe ce sommet à l'une de ces trois étiquettes : `hasLabel("Label1")` `hasLabel("Label2")` et `hasLabel("Label3")`.

Le délimiteur `::` est réservé à cet usage uniquement.

Vous ne pouvez pas spécifier plusieurs étiquettes dans l'étape `hasLabel`. Par exemple, `hasLabel("Label1::Label2")` ne correspond à rien.

16. Spécifiez Time/date.

```
g.V().property(single, 'lastUpdate', datetime('2018-01-01T00:00:00'))
```

Neptune ne prend pas en charge la date Java. Utilisez plutôt la `datetime()` fonction. `datetime()`accepte une ISO8061-compliant `datetime` chaîne de caractères.

Elle prend en charge les formats suivants : `YYYY-MM-DD, YYYY-MM-DDTHH:mm`, `YYYY-MM-DDTHH:mm:SS` et `YYYY-MM-DDTHH:mm:SSZ`.

17. Supprimez des sommets, des propriétés ou des arêtes.

```
g.V().hasLabel('person').properties('age').drop().iterate()
g.V('1').drop().iterate()
g.V().outE().hasLabel('created').drop()
```

Voici quelques exemples de suppression.

> **Note**
>
> L'étape `.next()` ne fonctionne pas avec `.drop()`. Utilisez `.iterate()` à la place.

18. Lorsque vous avez terminé, saisissez la commande suivante pour quitter la console Gremlin.

```
:exit
```

> **Note**
>
> Utilisez un point-virgule (`;`) ou un caractère de saut de ligne (\n) pour séparer chaque instruction.

Chaque traversée précédant la traversée finale doit se terminer par l'exécution de `iterate()`. Seules les données de la traversée finale sont renvoyées.

Utilisation d'OpenCypher pour accéder aux données graphiques dans Amazon Neptune

Pour commencer à utiliser OpenCypheropenCypher, consultez ou utilisez les blocs-notes OpenCypher du référentiel Neptune graph-notebook. GitHub

Utilisation de SPARQL pour accéder aux données graphiques dans Amazon Neptune

SPARQL est un langage de requête pour RDF (Resource Description Framework), qui est un format de données de graphe conçu pour le web. Amazon Neptune est compatible avec SPARQL 1.1. En d'autres termes, vous pouvez vous connecter à une instance de base de données Neptune et interroger le graphe à l'aide du langage de requête décrit dans la spécification SPARQL 1.1 Query Language.

Une requête dans SPARQL se compose d'une clause `SELECT` pour spécifier les variables à renvoyer et une clause `WHERE` clause pour spécifier les données de correspondance du graphe. Si vous ne connaissez pas les requêtes SPARQL, consultez Writing Simple Queries dans la section SPARQL 1.1 Query Language.

Le point de terminaison HTTP pour les requêtes SPARQL dans une instance de base de données Neptune est `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.

Pour vous connecter à SPARQL

1. Vous pouvez obtenir le point de terminaison SPARQL de votre cluster Neptune à partir de SparqlEndpointl'élément de la section Sorties de CloudFormation la pile.
2. Saisissez ce qui suit pour envoyer une requête SPARQL **`UPDATE`** à l'aide du protocole HTTP `POST` et de la commande curl.

```
curl -X POST --data-binary 'update=INSERT DATA { <https://test.com/s> <https://
test.com/p> <https://test.com/o> . }' https://your-neptune-endpoint:port/sparql
```

L'exemple précédent insère le triplet suivant dans le graphe SPARQL par défaut : `<https://test.com/s> <https://test.com/p> <https://test.com/o>`

3. Saisissez ce qui suit pour envoyer une requête SPARQL **`QUERY`** à l'aide du protocole HTTP `POST` et de la commande curl.

```
curl -X POST --data-binary 'query=select ?s ?p ?o where {?s ?p ?o} limit 10'
 https://your-neptune-endpoint:port/sparql
```

L'exemple précédent renvoie 10 triplets au plus (sujet-prédicat-objet) dans le graphe à l'aide de la requête `?s ?p ?o` avec une limite égale à 10. Pour interroger autre chose, remplacez la requête par une autre requête SPARQL.

> **Note**
>
> Le type MIME par défaut d'une réponse est `application/sparql-results+json` pour les requêtes `SELECT` et `ASK`.
> Le type MIME par défaut d'une réponse est `application/n-quads` pour les requêtes `CONSTRUCT` et `DESCRIBE`.
> Pour obtenir la liste des types MIME disponibles, consultez API HTTP SPARQL.

Chargement de données dans un cluster Amazon Neptune

Amazon Neptune fournit un processus pour le chargement de données directement dans une instance de base de données Neptune à partir de fichiers externes. Vous pouvez utiliser ce processus au lieu d'exécuter un grand nombre d'instructions `INSERT`, d'étapes `addV` et `addE` ou d'autres appels d'API.

Vous trouverez ci-dessous des liens vers des informations supplémentaires sur le chargement.

- Méthodes courantes de chargement de données : Chargement des données
- Formats de données pris en charge par le chargeur en bloc : the section called “Formats de données”
- Exemple de chargement : the section called “Exemple de chargement”

Surveillance d'un cluster Amazon Neptune

Amazon Neptune prend en charge les méthodes de surveillance suivantes.

- Amazon CloudWatch — Amazon Neptune envoie automatiquement des métriques aux alarmes CloudWatch et les prend également en charge CloudWatch . Pour de plus amples informations, veuillez consulter the section called “En utilisant CloudWatch”.
- AWS CloudTrail— Amazon Neptune prend en charge la journalisation des API en utilisant. CloudTrail Pour de plus amples informations, veuillez consulter the section called “Journalisation des appels d'API Neptune avec AWS CloudTrail”.
- Balisage – Utilisez des balises pour ajouter des métadonnées à vos ressources Neptune et suivre l'utilisation en fonction des balises. Pour de plus amples informations, veuillez consulter the section called “Marquage des ressources Neptune”.
- Fichiers journaux d'audit : affichez, téléchargez ou consultez les fichiers journaux de base de données à l'aide de la console Neptune. Pour de plus amples informations, veuillez consulter the section called “Journaux d'audit avec Neptune”.
- Statut de l'instance : vérifiez l'état du moteur de base de données orienté graphe d'une instance Neptune, déterminez la version du moteur qui est installée et obtenez d'autres informations sur le statut du moteur à l'aide de l'API de statut d'instance.

Résolution des problèmes liés à un cluster Amazon Neptune

Les liens suivants peuvent être vous utiles pour résoudre des problèmes liés à Amazon Neptune.

- Bonnes pratiques : pour trouver des solutions aux problèmes courants et des suggestions pour l'amélioration des performances, consultez .Bonnes pratiques
- Erreurs de service : pour obtenir une liste des erreurs concernant les API de gestion et les connexions à la base de données orientée graphe, consultez Erreurs Neptune.
- Limites de service : pour plus d'informations sur les limites Neptune, consultez Limites Neptune.
- Versions du moteur : pour plus d'informations sur les versions de moteur de graphe, y compris les notes de mise à jour, consultez Versions du moteur.
- Forums de support : pour participer à une discussion sur Neptune, consultez le Forum Amazon Neptune.
- Tarification : pour en savoir plus sur les coûts liés à l'utilisation d'Amazon Neptune, consultez Tarification Amazon Neptune.

- AWS Support — Pour obtenir de l'aide et des conseils de la part d'experts, consultez AWS Support.

Amazon Neptune sans serveur

Amazon Neptune sans serveur est une configuration de mise à l'échelle automatique à la demande conçue pour mettre à l'échelle la capacité de votre cluster de bases de données en fonction des besoins, afin de répondre à des augmentations même très importantes des besoins de traitement, puis de la réduire à nouveau lorsque les besoins diminuent. Il contribue à automatiser les processus de surveillance de la charge de travail et d'ajustement de la capacité de la base de données Neptune. La capacité est automatiquement ajustée en fonction des besoins de l'application, et seules les ressources consommées par votre application vous sont facturées.

Cas d'utilisation de Neptune sans serveur

Neptune sans serveur prend en charge de nombreux types de charges de travail. Il convient aux charges de travail exigeantes et très variables et peut être très utile si l'utilisation de votre base de données est généralement intensive pendant de courtes périodes, suivies de longues périodes avec une activité légère, voire pas d'activité du tout. Neptune sans serveur est particulièrement utile pour les cas d'utilisation suivants :

- Charges de travail variables : charges de travail présentant des hausses soudaines et imprévisibles de l'activité du CPU. Avec Neptune sans serveur, la capacité de votre base de données orientée graphe est automatiquement augmentée pour répondre aux besoins de la charge de travail, puis elle est réduite à nouveau lorsque le pic d'activité est terminé. Vous n'avez plus besoin de provisionner pour un pic de capacité ou une capacité moyenne. Vous pouvez spécifier une limite de capacité supérieure pour faire face aux pics d'utilisation des charges de travail. Cette capacité n'est utilisée que si elle est nécessaire.

  La précision de la mise à l'échelle fournie par Neptune sans serveur vous permet de faire correspondre étroitement la capacité aux besoins de votre charge de travail. Neptune sans serveur peut ajouter ou supprimer de la capacité par paliers précis en fonction des besoins. Il peut n'ajouter que la moitié d'une unité de capacité Neptune (NCU) lorsque vous n'avez besoin que d'un peu plus de capacité.
- Multi-tenant applications — En tirant parti de Neptune Serverless, vous pouvez créer un cluster de base de données distinct pour chacune des applications que vous devez exécuter sans avoir à gérer ces clusters locataires individuellement. Chacun des clusters locataires peut avoir des périodes d'activité et d'inactivité différentes en fonction de plusieurs facteurs, mais Neptune sans serveur peut les mettre à l'échelle efficacement sans votre intervention.

- Nouvelles applications : lorsque vous déployez une nouvelle application, vous n'êtes souvent pas certain de la capacité de base de données dont elle aura besoin. Neptune sans serveur vous permet de configurer un cluster de bases de données capable de se mettre à l'échelle automatiquement pour répondre aux exigences de capacité des nouvelles applications au fur et à mesure de leur développement.
- Planification de la capacité : supposons que vous ajustiez généralement la capacité de votre base de données (ou que vous vérifiiez la capacité de base de données optimale pour votre charge de travail) en modifiant les classes de toutes les instances de base de données d'un cluster. Avec Neptune sans serveur, vous pouvez éviter cette surcharge administrative. Au lieu de cela, vous pouvez remplacer les instances de base de données provisionnées par des instances sans serveur ou inversement, sans avoir à créer un nouveau cluster de bases de données ou une nouvelle instance de base de données.
- Développement et tests : Neptune sans serveur convient également parfaitement aux environnements de développement et de test. Avec Neptune sans serveur, vous pouvez créer des instances de base de données avec une capacité maximale suffisamment élevée pour tester l'application la plus exigeante, et une faible capacité minimale pour tous les autres cas où le système peut être inactif entre les tests.

Neptune sans serveur met uniquement à l'échelle la capacité de calcul. Votre volume de stockage reste le même et n'est pas affecté par la mise à l'échelle sans serveur.

> **Note**
>
> Vous pouvez également utiliser l'auto-scaling Neptune avec Neptune sans serveur pour gérer différents types de variantes de charge de travail.

Contraintes d'Amazon Neptune sans serveur

- Neptune Serverless est uniquement disponible dans les régions suivantes :
  - USA Est (Virginie du Nord) : `us-east-1`
  - USA Est (Ohio) : `us-east-2`
  - USA Ouest (Californie du Nord) : `us-west-1`
  - USA Ouest (Oregon) : `us-west-2`
  - Canada (Centre) : `ca-central-1`

- Europe (Stockholm) : `eu-north-1`
- Europe (Espagne) : `eu-south-2`
- Europe (Irlande) : `eu-west-1`
- Europe (Londres) : `eu-west-2`
- Europe (Paris) : `eu-west-3`
- Europe (Francfort) : `eu-central-1`
- Europe (Zurich) : `eu-central-2`
- Asie-Pacifique (Tokyo) : `ap-northeast-1`
- Asie-Pacifique (Séoul) : `ap-northeast-2`
- Asie-Pacifique (Singapour) : `ap-southeast-1`
- Asie-Pacifique (Sydney) : `ap-southeast-2`
- Asie-Pacifique (Jakarta) : `ap-southeast-3`
- Asie-Pacifique (Hong Kong) : `ap-east-1`
- Asie-Pacifique (Mumbai) : `ap-south-1`
- Asie-Pacifique (Hyderabad) : `ap-south-2`
- Amérique du Sud (São Paulo) : `sa-east-1`

- Pas disponible dans les premières versions du moteur : Neptune sans serveur n'est disponible que dans les versions 1.2.0.1 ou ultérieures du moteur.
- Pas compatible avec le cache de recherche Neptune : le cache de recherche ne fonctionne pas avec les instances de base de données sans serveur.
- La mémoire maximale d'une instance sans serveur est de 256 Go : définir `MaxCapacity` sur 128 NCU (le paramètre le plus élevé pris en charge) permet à une instance Neptune sans serveur de passer à 256 Go de mémoire, ce qui est équivalent à un type d'instance `R6g.8XL` provisionné.

Mise à l'échelle de la capacité dans un cluster de bases de données Neptune sans serveur

La configuration d'un cluster de bases de données Neptune sans serveur est similaire à la configuration d'un cluster standard provisionné, avec une configuration supplémentaire pour les unités minimales et maximales correspondant à la mise à l'échelle, et avec le type d'instance défini sur `db.serverless`. La configuration de la mise à l'échelle est définie en unités de capacité Neptune (NCU), dont chacune comprend 2 Gio (gibioctet) de mémoire (RAM) en plus de la capacité

du processeur virtuel (vCPU) et du réseau associés. Elle est définie en tant que partie d'un objet `ServerlessV2ScalingConfiguration`, représenté en JSON comme ceci :

```
"ServerlessV2ScalingConfiguration": {
  "MinCapacity": (minimum NCUs, a floating-point number such as   1.0),
  "MaxCapacity": (maximum NCUs, a floating-point number such as 128.0)
}
```

À tout moment, chaque instance d'enregistreur ou de lecteur Neptune a une capacité mesurée par un nombre à virgule flottante qui représente le nombre de NCU actuellement utilisées par cette instance. Vous pouvez utiliser la CloudWatch ServerlessDatabaseCapacitymétrique au niveau de l'instance pour connaître le nombre de NCU qu'une instance de base de données donnée utilise actuellement, et la métrique NCUUtilization pour déterminer le pourcentage de sa capacité maximale utilisée par l'instance. Ces deux métriques sont également disponibles au niveau d'un cluster de bases de données pour indiquer l'utilisation moyenne de ses ressources dans son ensemble.

Lorsque vous créez un cluster de bases de données Neptune sans serveur, vous définissez le nombre minimal et maximal d'unités de capacité Neptune (NCU) pour toutes les instances sans serveur.

La valeur NCU minimale que vous spécifiez définit la plus petite taille possible d'une instance sans serveur du cluster de bases de données. De même, la valeur NCU maximale définit la plus grande taille possible d'une instance sans serveur. La valeur NCU maximale que vous pouvez définir est de 128 NCU, tandis que la valeur minimale est de 1 NCU.

Neptune suit en permanence la charge de chaque instance Neptune sans serveur en surveillant son utilisation de ressources telles que le CPU, la mémoire et le réseau. La charge est générée par les opérations de base de données de votre application, par le traitement en arrière-plan pour le serveur et par d'autres tâches administratives.

Lorsque la charge d'une instance sans serveur atteint la limite de capacité actuelle ou lorsque Neptune détecte d'autres problèmes de performance, la capacité de l'instance augmente automatiquement. Lorsque la charge de l'instance baisse, la capacité diminue pour atteindre les unités de capacité minimale configurées: La capacité du CPU est libérée avant la mémoire. Cette architecture permet de libérer les ressources de manière progressive et contrôlée et de gérer efficacement les fluctuations de la demande.

Vous pouvez faire en sorte qu'une instance de lecteur soit mise à l'échelle en même temps que l'instance d'enregistreur ou indépendamment en définissant son niveau de promotion. Les instances de lecteur des niveaux de promotion 0 et 1 sont mis à l'échelle en même temps que

l'enregistreur. Cela assure un dimensionnement correct de la capacité permettant de prendre en charge la charge de travail de l'enregistreur rapidement en cas de basculement. Les lecteurs des niveaux de promotion 2 à 15 sont mis à l'échelle indépendamment de l'instance d'enregistreur et indépendamment des autres lecteurs.

Si vous avez créé votre cluster de base de données Neptune en tant que Multi-AZ cluster pour garantir une haute disponibilité, Neptune Serverless fait évoluer les instances dans toutes les zones de disponibilité à la hausse ou à la baisse en fonction de la charge de votre base de données. Vous pouvez définir le niveau de promotion d'une instance de lecteur dans une zone de disponibilité secondaire sur 0 ou 1 afin que sa capacité augmente ou diminue en fonction de la capacité de l'instance d'enregistreur dans la zone de disponibilité principale et afin qu'elle puisse prendre en charge la charge de travail actuelle à tout moment.

> **Note**
>
> Le stockage d'un cluster de base de données Neptune consiste en six copies de toutes vos données, réparties sur trois AZ, que vous ayez créé le cluster en tant que Multi-AZ cluster ou non. La réplication du stockage est gérée par le sous-système de stockage et n'est pas concernée par Neptune sans serveur.

Choix d'une valeur de capacité minimale pour un cluster de bases de données Neptune sans serveur

La plus petite valeur que vous pouvez définir pour la capacité minimale est `1.0` NCU.

Veillez à ne pas définir une valeur minimale inférieure à ce dont votre application a besoin pour fonctionner efficacement. Une valeur trop faible peut entraîner un taux d'expiration plus élevé pour certaines charges de travail gourmandes en mémoire.

En définissant une valeur minimale aussi basse que possible, vous réalisez des écononomies, car le cluster utilisera un minimum de ressources lorsque la demande sera faible. Toutefois, si votre charge de travail a tendance à fluctuer considérablement, en passant d'une capacité très faible à très élevée, vous pouvez fixer le minimum à un niveau plus élevé afin de pouvoir augmenter la capacité des instances Neptune sans serveur plus rapidement.

Cela est dû au fait que Neptune choisit les incréments de mise à l'échelle en fonction de la capacité actuelle. Si la capacité actuelle est faible, Neptune augmente lentement la capacité dans un premier temps. Si la capacité minimale est plus élevée, Neptune commence par un incrément de mise

à l'échelle plus grand et peut donc augmenter plus rapidement la capacité pour faire face à une augmentation soudaine et importante de la charge de travail.

Choix d'une valeur de capacité maximale pour un cluster de bases de données Neptune sans serveur

La valeur la plus élevée que vous pouvez définir pour la capacité maximale est `128.0` NCU, tandis que la valeur la plus faible que vous pouvez définir pour la capacité maximale est `2.5` NCU. Quelle que soit la valeur de capacité maximale que vous définissez, elle doit être au moins aussi élevée que la valeur de capacité minimale que vous avez spécifiée.

En règle générale, définissez une valeur maximale suffisamment élevée pour gérer les pics de charge que votre application est susceptible de rencontrer. Une valeur trop faible peut entraîner un taux d'expiration plus élevé pour certaines charges de travail gourmandes en mémoire.

Définir une valeur maximale aussi élevée que possible présente l'avantage de permettre à votre application de gérer les charges de travail les plus inattendues. L'inconvénient est qu'il devient plus difficile de prévoir et de contrôler les coûts des ressources. Une hausse inattendue de la demande peut finir par coûter beaucoup plus cher que ce que votre budget avait prévu.

L'avantage d'une valeur maximale soigneusement ciblée est qu'elle vous permet de répondre à toute augmentation soudaine de la demande tout en plafonnant les coûts de calcul Neptune.

> **Note**
>
> La modification de la plage de capacité d'un cluster de bases de données Neptune sans serveur modifie les valeurs par défaut de certains paramètres de configuration. Neptune peut appliquer certaines de ces nouvelles valeurs par défaut immédiatement, mais certaines modifications de paramètre dynamiques ne prennent effet qu'après un redémarrage. Un statut `pending-reboot` indique qu'un redémarrage est nécessaire pour appliquer certaines modifications de paramètre.

Utilisation de votre configuration existante pour estimer les besoins sans serveur

En règle générale, si vous modifiez la classe de vos instances de base de données provisionnées pour gérer une charge de travail particulièrement élevée ou faible, vous pouvez utiliser cette

expérience pour effectuer une estimation approximative de la plage de capacité Neptune sans serveur équivalente.

Estimation du paramètre de capacité minimale le plus approprié

Vous pouvez appliquer ce que vous savez de votre cluster de bases de données Neptune existant pour estimer le paramètre de capacité minimale sans serveur qui fonctionnera le mieux.

Par exemple, si votre charge de travail provisionnée présente des exigences en mémoire trop élevées pour les petites classes d'instances de base de données telles que `T3` ou `T4g`, choisissez un nombre minimal de NCU qui fournit une quantité de mémoire comparable à une instance de base de données `R5` ou `R6g`.

Supposons que vous utilisiez la classe d'instances de base de données `db.r6g.xlarge` lorsque la charge de travail de votre cluster est faible. Cette classe d'instances de base de données dispose de 32 Gio de mémoire. Vous pouvez donc spécifier 16 NCU comme paramètre de capacité minimale pour créer des instances sans serveur pouvant réduire leur capacité en conséquence (chaque NCU correspond à environ 2 Gio de mémoire environ). Si l'instance `db.r6g.xlarge` est parfois sous-exploitée, vous pourriez même spécifier une valeur inférieure.

Si votre application fonctionne le plus efficacement lorsque vos instances de base de données peuvent contenir une quantité spécifique de données en mémoire ou dans le cache de mémoire tampon, envisagez de spécifier un nombre minimal de NCU suffisamment grand pour fournir suffisamment de mémoire à cet effet. Dans le cas contraire, les données risquent d'être expulsées du cache tampon lorsque la capacité des instances sans serveur baissera, et devront être ramenées dans le cache de la mémoire tampon au fil du temps lorsque la capacité des instances augmentera. Si la quantité de données I/O à ramener dans le cache tampon est importante, il peut être intéressant de choisir une valeur NCU minimale plus élevée.

Si vous constatez que vos instances sans serveur fonctionnent la plupart du temps à une capacité spécifique, il est judicieux de définir une capacité minimale légèrement inférieure à celle-ci. Neptune sans serveur estime efficacement dans quelle mesure et avec quelle rapidité procéder à l'augmentation d'échelle lorsque la capacité actuelle n'est pas nettement inférieure à la capacité requise.

Si vous disposez d'un cluster à configuration mixte avec un enregistreur provisionné et des lecteurs Neptune sans serveur, les lecteurs ne sont pas mis à l'échelle avec l'enregistreur. Comme ils sont mis à l'échelle de façon indépendante, leur attribuer une valeur de capacité minimale faible peut entraîner un retard de réplication excessif. Ils ne disposeront peut-être pas de la capacité suffisante

pour suivre les modifications apportées par l'enregistreur lorsque la charge de travail implique un très grand nombre d'écritures. Dans ce cas, définissez une capacité minimale comparable à la capacité de l'enregistreur. Si vous constatez un retard de réplica dans les lecteurs aux niveaux de promotion 2 à 15, augmentez la valeur de capacité minimale du cluster.

Estimation du paramètre de capacité maximale le plus approprié

Vous pouvez appliquer ce que vous savez de votre cluster de bases de données Neptune existant pour estimer le paramètre de capacité maximale sans serveur qui fonctionnera le mieux.

Par exemple, supposons que vous utilisiez la classe d'instances de base de données `db.r6g.4xlarge` lorsque la charge de travail de votre cluster est élevée. Cette classe d'instances de base de données dispose de 128 Gio de mémoire. Vous pouvez donc spécifier 64 NCU comme paramètre maximal pour configurer des instances Neptune sans serveur équivalentes (chaque NCU correspond à environ 2 Gio de mémoire). Vous pouvez spécifier une valeur plus élevée pour augmenter davantage la capacité de l'instance de base de données au cas où votre instance `db.r6g.4xlarge` ne pourrait pas toujours gérer la charge de travail.

Si les pics inattendus de votre charge de travail sont rares, il peut être judicieux de définir une capacité maximale suffisamment élevée pour maintenir les performances des applications même pendant ces pics. D'autre part, vous souhaiterez peut-être définir une capacité maximale inférieure qui peut réduire le débit lors des pics inhabituels, tout en permettant à Neptune de gérer les charges de travail habituelles sans problème et en limitant les coûts.

Configuration supplémentaire pour les instances et les clusters de bases de données Neptune sans serveur

Outre la définition de la capacité minimale et maximale du cluster de bases de données Neptune sans serveur, vous devez prendre en compte quelques autres choix de configuration.

Combinaison d'instances sans serveur et d'instances provisionnées dans un cluster de bases de données

Un cluster de bases de données ne doit pas nécessairement être uniquement sans serveur : vous pouvez créer une combinaison d'instances sans serveur et d'instances provisionnées (configuration mixte).

Supposons, par exemple, que vous ayez besoin d'une capacité d'écriture supérieure à celle disponible pour une instance sans serveur. Dans ce cas, vous pouvez configurer le cluster avec un

enregistreur provisionné de très grande taille tout en utilisant des instances sans serveur pour les lecteurs.

Supposons également que la charge de travail d'écriture de votre cluster varie, mais que la charge de travail de lecture soit stable. Dans ce cas, vous pouvez configurer le cluster avec un enregistreur sans serveur et un ou plusieurs lecteurs provisionnés.

Pour plus d'informations sur la création d'un cluster de bases de données à configuration mixte, consultez Utilisation d'Amazon Neptune sans serveur.

Définition des niveaux de promotion des instances Neptune sans serveur

Pour les clusters contenant plusieurs instances sans serveur ou une combinaison d'instances sans serveur et d'instances provisionnées, prêtez attention au paramètre de niveau de promotion pour chaque instance sans serveur. Ce paramètre contrôle davantage le comportement des instances sans serveur que celui des instances de base de données provisionnées.

Dans le AWS Management Console, vous spécifiez ce paramètre en utilisant la priorité Failover sous Configuration supplémentaire sur les pages Créer une base de données, Modifier une instance et Ajouter un lecteur. Cette propriété s'affiche pour les instances de base de données existantes dans la colonne facultative Niveau de priorité sur la page Bases de données. Cette propriété est également disponible sur la page de détails d'un cluster de bases de données ou d'une instance.

Pour les instances provisionnées, le choix du niveau 0 à 15 détermine uniquement l'ordre dans lequel Neptune choisit l'instance de lecteur à promouvoir en enregistreur lors d'une opération de basculement. Pour les instances de lecteur Neptune sans serveur, le numéro de niveau détermine également si la capacité de l'instance augmente pour correspondre à celle de l'instance d'enregistreur ou si elle se met à l'échelle indépendamment de celle-ci en fonction uniquement de sa propre charge de travail.

Les instances de lecteur Neptune sans serveur de niveau 0 ou 1 sont maintenues à une capacité minimale au moins égale à celle de l'instance d'enregistreur afin d'être prêtes à prendre le relais de l'enregistreur en cas de basculement. Si l'enregistreur est une instance provisionnée, Neptune estime la capacité sans serveur équivalente et utilise cette estimation comme capacité minimale pour l'instance de lecteur sans serveur.

Les instances de lecteur Neptune sans serveur des niveaux 2 à 15 n'ont pas la même contrainte de capacité minimale et peuvent faire l'objet d'une mise à l'échelle indépendamment de l'enregistreur. Lorsqu'elles sont inactives, leur capacité est réduite à la valeur NCU minimale spécifiée dans la plage

de capacité du cluster. Cela peut toutefois poser des problèmes si la charge de travail de lecture fait l'objet d'une hausse soudaine.

Assurer l'alignement entre la capacité des lecteurs et la capacité de l'enregistreur

Il est important de vous assurer que les instances de lecteur peuvent suivre le rythme de l'instance d'enregistreur, afin d'éviter un retard de réplication excessif. Cela est particulièrement préoccupant dans deux situations, où les instances de lecteur sans serveur ne sont pas automatiquement mises à l'échelle en synchronisation avec l'instance d'enregistreur :

- Lorsque l'enregistreur est provisionné et que les lecteurs sont sans serveur.
- Lorsque l'enregistreur est sans serveur et que les lecteurs sans serveur sont aux niveaux de promotion 2 à 15.

Dans les deux cas, définissez la capacité minimale sans serveur pour qu'elle corresponde à la capacité d'enregistreur attendue, afin de garantir que les opérations de lecture n'expirent pas et ne provoquent pas de redémarrages potentiels. Dans le cas d'une instance d'enregistreur provisionnée, définissez la capacité minimale pour qu'elle corresponde à celle de l'instance provisionnée. Dans le cas d'un enregistreur sans serveur, le paramètre optimal peut être plus difficile à prévoir.

Comme la plage de capacité des instances est définie au niveau du cluster, toutes les instances sans serveur sont contrôlées par les mêmes paramètres de capacité minimale et maximale. Les instances de lecteur des niveaux 0 et 1 se mettent à l'échelle en synchronisation avec l'instance d'enregistreur, mais les instances des niveaux de promotion 2 à 15 se mettent à l'échelle indépendamment les unes des autres et de l'instance d'enregistreur, en fonction de leur charge de travail. Si vous définissez une capacité minimale trop faible, la capacité des instances inactives des niveaux 2 à 15 risque d'être trop réduite pour pouvoir être réaugmentée assez rapidement afin de faire face à une augmentation soudaine de l'activité de l'enregistreur.

Éviter de définir une valeur de délai d'expiration trop élevée

Il est possible que vous encouriez des coûts inattendus si vous définissez une valeur trop élevée pour le délai d'expiration des requêtes sur une instance sans serveur.

Sans paramètre de délai d'expiration raisonnable, vous risquez d'émettre par inadvertance une requête qui nécessite un type d'instance puissant et coûteux et qui continue de s'exécuter pendant très longtemps, entraînant ainsi des coûts que vous n'auriez jamais anticipés. Pour éviter cette

situation, utilisez une valeur de délai d'expiration qui convient à la plupart des requêtes et qui n'implique que l'expiration de celles dont le délai est étonnamment long.

Ce principe est valable à la fois pour les valeurs de délai d'expiration générales définies à l'aide de paramètres et pour les valeurs de délai d'expiration par requête définies à l'aide d'indicateurs de requête.

Optimisation de la configuration de Neptune sans serveur

Si le cluster de bases de données Neptune sans serveur n'est pas adapté à la charge de travail qu'il exécute, vous remarquerez peut-être qu'il ne fonctionne pas de manière optimale. Vous pouvez ajuster le paramètre de capacité and/or maximale minimale afin qu'il puisse évoluer sans rencontrer de problèmes de mémoire.

- Augmentez la valeur du paramètre de capacité minimale du cluster. Cette action peut remédier à la situation dans laquelle une instance inactive revient à une capacité qui dispose de moins de mémoire que ce dont votre application et les fonctionnalités activées ont besoin.
- Augmentez la valeur du paramètre de capacité maximale du cluster. Cette action peut remédier à la situation dans laquelle la capacité d'une base de données occupée ne parvient pas à augmenter au point d'atteindre une capacité où la mémoire serait suffisante pour gérer la charge de travail ainsi que les fonctionnalités qui ont été activées et qui nécessitent beaucoup de mémoire.
- Modifiez la charge de travail sur l'instance en question. Par exemple, vous pouvez ajouter des instances de lecteur au cluster afin de répartir la charge sur d'autres instances.
- Ajustez les requêtes de votre application afin qu'elles utilisent moins de ressources.
- Essayez d'utiliser une instance provisionnée dont la taille est supérieure au nombre maximal de NCU disponibles dans Neptune sans serveur, afin de déterminer si elle répond mieux aux exigences de mémoire et de CPU de la charge de travail.

Utilisation d'Amazon Neptune sans serveur

Vous pouvez créer un cluster de bases de données Neptune en tant que cluster sans serveur ou, dans certains cas, vous pouvez convertir un cluster de bases de données existant en cluster sans serveur. Vous pouvez également convertir les instances de base de données d'un cluster de bases de données sans serveur en instances sans serveur, et inversement. Vous ne pouvez utiliser Neptune Serverless que dans les pays Régions AWS où il est pris en charge, avec quelques autres limitations (voir). Contraintes d'Amazon Neptune sans serveur

Vous pouvez également utiliser la pile Neptune CloudFormation pour créer un cluster de bases de données Neptune sans serveur.

Création d'un cluster de bases de données utilisant le mode sans serveur

Pour créer un cluster de bases de données Neptune utilisant le mode sans serveur, vous pouvez le faire en utilisant la AWS Management Console de la même manière que pour créer un cluster provisionné. La différence est que sous Taille de l'instance de base de données, vous devez définir la classe d'instances de base de données en mode sans serveur. Lorsque vous effectuez cette opération, vous devez définir la plage de capacité sans serveur du cluster.

Vous pouvez également créer un cluster de base de données sans serveur à l'aide des commandes AWS CLI with comme celle-ci (sous Windows, remplacez « \ » par « ^ ») :

```
aws neptune create-db-cluster \
  --region (an Région AWS region that supports serverless) \
  --db-cluster-identifier (ID for the new serverless DB cluster) \
  --engine neptune \
  --engine-version (optional: 1.2.0.1 or above) \
  --serverless-v2-scaling-configuration "MinCapacity=1.0, MaxCapacity=128.0"
```

Vous pouvez également spécifier le paramètre `serverless-v2-scaling-configuration` comme suit :

```
  --serverless-v2-scaling-configuration '{"MinCapacity":1.0, "MaxCapacity":128.0}'
```

Vous pouvez ensuite exécuter la commande `describe-db-clusters` pour l'attribut `ServerlessV2ScalingConfiguration`, qui devrait renvoyer les paramètres de plage de capacité que vous avez spécifiés :

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": (the specified minimum number of NCUs),
    "MaxCapacity": (the specified maximum number of NCUs)
}
```

Conversion d'un cluster de bases de données ou d'une instance de base de données existant en mode sans serveur

Si vous avez un cluster de bases de données Neptune qui utilise la version 1.2.0.1 ou supérieure du moteur, vous pouvez le convertir en cluster sans serveur. Ce processus entraîne des temps d'arrêt.

La première étape consiste à ajouter une plage de capacité au cluster existant. Vous pouvez le faire en utilisant AWS Management Console le ou en utilisant une AWS CLI commande comme celle-ci (sous Windows, remplacez « \ » par « ^ ») :

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (your DB cluster ID) \
  --serverless-v2-scaling-configuration \
      MinCapacity=(minimum number of NCUs, such as  2.0), \
      MaxCapacity=(maximum number of NCUs, such as 24.0)
```

L'étape suivante consiste à créer une instance de base de données sans serveur pour remplacer l'instance principale existante (enregistreur) dans le cluster. Encore une fois, vous pouvez effectuer cette opération et toutes les étapes suivantes en utilisant le AWS Management Console ou le AWS CLI. Dans les deux cas, spécifiez la classe d'instances de base de données comme étant sans serveur. La AWS CLI commande ressemblerait à ceci (sous Windows, remplacez « \ » par « ^ ») :

```
aws neptune create-db-instance \
  --db-instance-identifier (an instance ID for the new writer instance) \
  --db-cluster-identifier (ID of the DB cluster) \
  --db-instance-class db.serverless \
  --engine neptune
```

Lorsque la nouvelle instance d'enregistreur est disponible, effectuez un basculement pour en faire l'instance d'enregistreur du cluster :

```
aws neptune failover-db-cluster \
  --db-cluster-identifier (ID of the DB cluster) \
  --target-db-instance-identifier (instance ID of the new serverless instance)
```

Supprimez ensuite l'ancienne instance d'enregistreur :

```
aws neptune delete-db-instance \
```

```
  --db-instance-identifier (instance ID of the old writer instance) \
  --skip-final-snapshot
```

Enfin, faites de même pour créer une instance sans serveur qui remplacera chaque instance de lecteur provisionnée que vous souhaitez convertir en instance sans serveur, et supprimez les instances provisionnées existantes (aucun basculement n'est nécessaire pour les instances de lecteur).

Modification de la plage de capacité d'un cluster de bases de données sans serveur

Vous pouvez modifier la plage de capacité d'un cluster de bases de données Neptune sans serveur en utilisant l' AWS CLI comme suit (sous Windows, remplacez « \ » par « ^ ») :

```
aws neptune modify-db-cluster \
  --region (an AWS region that supports serverless) \
  --db-cluster-identifier (ID of the serverless DB cluster) \
  --apply-immediately \
  --serverless-v2-scaling-configuration MinCapacity=4.0, MaxCapacity=32
```

La modification de la plage de capacité modifie les valeurs par défaut de certains paramètres de configuration. Neptune peut appliquer certaines de ces nouvelles valeurs par défaut immédiatement, mais certaines modifications de paramètre dynamiques ne prennent effet qu'après un redémarrage. Le statut `pending-reboot` indique qu'un redémarrage est nécessaire pour appliquer certaines modifications de paramètre.

Conversion d'une instance de base de données sans serveur en instance provisionnée

Pour convertir une instance Neptune sans serveur en instance provisionnée, il vous suffit de remplacer sa classe d'instances par l'une des classes d'instances provisionnées. Consultez Modification d'une instance de base de données Neptune (et application immédiate).

Configuration des clients G705 pour Serverless

Lorsque vous utilisez des WebSocket clients G705 avec Neptune Serverless, vous devez configurer l'intervalle de pulsation du client de manière appropriée afin de maintenir des connexions stables pendant les événements de dimensionnement. Pour obtenir des instructions de configuration

détaillées pour les clients Java JavaScript/Node.js, Go et Python, consultezConfiguration Heartbeat pour Neptune Serverless.

Surveillance de la capacité sans serveur avec Amazon CloudWatch

Vous pouvez l'utiliser CloudWatch pour surveiller la capacité et l'utilisation des instances sans serveur Neptune dans votre cluster de base de données. Deux CloudWatch mesures vous permettent de suivre la capacité sans serveur actuelle à la fois au niveau du cluster et au niveau de l'instance :

- **`ServerlessDatabaseCapacity`** : en tant que métrique au niveau de l'instance, `ServerlessDatabaseCapacity` indique la capacité actuelle de l'instance, en NCU. En tant que métrique au niveau du cluster, elle représente la moyenne de toutes valeurs `ServerlessDatabaseCapacity` de toutes les instances de base de données du cluster.
- **`NCUUtilization`** : cette métrique indique le pourcentage de capacité possible utilisée. Elle est calculée en utilisant la valeur `ServerlessDatabaseCapacity` actuelle (au niveau de l'instance ou du cluster) divisée par le paramètre de capacité maximale du cluster de bases de données.

  Si cette métrique approche les 100 % au niveau d'un cluster, ce qui signifie que la capacité du cluster est aussi élevée qu'elle peut l'être, envisagez d'augmenter le paramètre de capacité maximale.

  Si elle approche les 100 % pour une instance de lecteur alors que l'instance d'enregistreur n'est pas proche de sa capacité maximale, envisagez d'ajouter d'autres instances de lecteur pour répartir la charge de travail de lecture.

Notez que les métriques `CPUUtilization` et `FreeableMemory` ont des significations légèrement différentes pour les instances sans serveur et pour les instances provisionnées. Dans un contexte sans serveur, `CPUUtilization` est un pourcentage calculé comme la quantité de CPU actuellement utilisée, divisée par la quantité de CPU disponible à la capacité maximale. De même, `FreeableMemory` indique la quantité de mémoire libérable qui serait disponible si une instance était à sa capacité maximale.

L'exemple suivant montre comment utiliser AWS CLI le sous Linux pour récupérer les valeurs de capacité minimale, maximale et moyenne pour une instance de base de données donnée, mesurées toutes les 10 minutes pendant une heure. La commande Linux `date` indique les heures de début et de fin par rapport à la date et à l'heure actuelles. La fonction `sort_by` du paramètre `--query` trie les résultats par ordre chronologique en fonction du champ `Timestamp` :

```
aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '1 hour ago')" \
  --end-time "$(date -d 'now')" \
  --period 600 \
  --namespace "AWS/Neptune"
  --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=(instance ID) \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
  --output table
```

Utilisation d'Amazon Neptune avec une base de données mondiale

Une base de données mondiale Amazon Neptune couvre plusieurs bases de données Régions AWS, ce qui permet des lectures globales à faible latence et une restauration rapide dans les rares cas où une panne affecte l'ensemble d'une base de données. Région AWS

Une base de données Neptune globale se compose d'un cluster de bases de données principal dans une région et de jusqu'à cinq clusters de bases de données secondaires dans différentes régions.

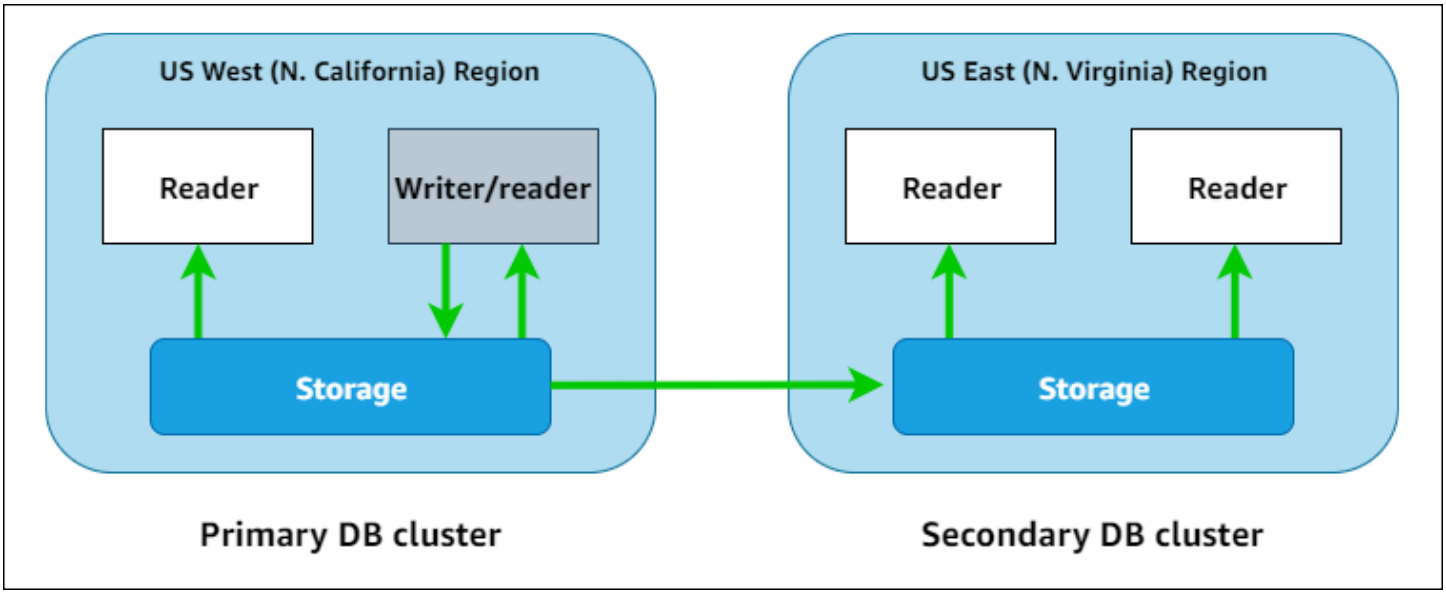
Les écritures ne peuvent avoir lieu que dans la région principale. Les régions secondaires ne prennent en charge que les lectures. Chaque région secondaire peut comporter jusqu'à 16 instances de lecteur.

Bases de données mondiales dans Amazon Neptune

En utilisant une base de données Neptune globale, vous pouvez exécuter vos applications distribuées dans le monde entier à l'aide d'une base de données unique couvrant plusieurs régions Régions AWS.

Une base de données Neptune globale se compose d'un cluster de bases de données principal dans une Région AWS principale et de jusqu'à cinq clusters de bases de données en lecture seule dans des Régions AWS secondaires. Lorsque vous effectuez une opération d'écriture dans le cluster de bases de données principal, Neptune réplique les données écrites dans tous les clusters de bases de données secondaires à l'aide d'une infrastructure dédiée, avec une latence généralement inférieure à une seconde.

Le schéma suivant montre un exemple de base de données globale qui comprend deux Régions AWS bases de données :



Vous pouvez mettre à l'échelle chaque cluster secondaire indépendamment pour gérer les charges de travail en lecture seule en ajoutant une ou plusieurs instances de réplica en lecture.

Pour effectuer des opérations d'écriture, vous devez vous connecter au point de terminaison du cluster de bases de données principal. Seul le cluster principal peut exécuter les opérations d'écriture. Ensuite, comme le montre le schéma ci-dessus, la réplication est effectuée par le volume de stockage du cluster, et non par le moteur de base de données.

Les bases de données Neptune globales sont conçues pour les applications ayant une empreinte mondiale. Les clusters de bases de données secondaires en lecture seule prennent en charge les opérations de lecture au plus près des utilisateurs de l'application.

Une base de données Neptune globale permet deux approches différentes de basculement.

- Pour effectuer une reprise après une panne dans la région principale, utilisez le detach-and-promote processus manuel non planifié, qui consiste à détacher l'un des clusters secondaires, à le transformer en cluster autonome, puis à le promouvoir en tant que nouveau cluster principal.
- Pour les procédures opérationnelles planifiées telles que la maintenance, utilisez le basculement planifié géré, dans lequel vous déplacez le cluster principal vers l'une de ses régions secondaires sans perte de données.

Avantages liés à l'utilisation de bases de données globales dans Amazon Neptune

En utilisant une base de données Neptune globale, vous bénéficiez des avantages suivants :

- Lectures globales avec latence locale : si vous avez des bureaux répartis dans le monde entier, une base de données globale permet à ceux qui se trouvent dans les régions secondaires d'accéder aux données dans leur propre région avec une latence locale.
- Clusters de bases de données Neptune secondaires évolutifs : vous pouvez mettre à l'échelle les clusters secondaires en ajoutant d'autres instances de réplicas en lecture seule. Comme les clusters secondaires sont en lecture seule, ils peuvent chacun prendre en charge jusqu'à 16 réplicas en lecture plutôt que la limite habituelle qui s'élève à 15 réplicas.
- Réplication rapide vers les clusters de bases de données secondaires : la réplication des clusters de bases de données principaux vers les clusters secondaires est rapide, avec une latence généralement en dessous d'une seconde et un impact moindre sur les performances du cluster de bases de données principal. La réplication étant effectuée au niveau du stockage, les ressources de l'instance de base de données sont entièrement disponibles pour les charges de travail de lecture et d'écriture des applications.
- Récupération suite aux pannes à l'échelle de la région : les clusters de bases de données secondaires vous permettent de déplacer une base de données globale vers une nouvelle région plus rapidement, avec un RTO moins élevé et avec moins de perte de données (RPO plus faible) que les solutions de réplication traditionnelles.

Limitations des bases de données globales Amazon Neptune

Les limitations suivantes s'appliquent actuellement aux bases de données globales :

- Les bases de données globales Neptune sont disponibles dans les Régions AWS suivantes :
  - USA Est (Virginie du Nord) : `us-east-1`
  - USA Est (Ohio) : `us-east-2`
  - USA Ouest (Californie du Nord) : `us-west-1`
  - USA Ouest (Oregon) : `us-west-2`
  - Canada-Ouest (Calgary) : `ca-west-1`
  - Europe (Espagne) : `eu-south-2`

- Europe (Irlande) : `eu-west-1`
- Europe (Londres) : `eu-west-2`
- Europe (Francfort) : `eu-central-1`
- Asie-Pacifique (Tokyo) : `ap-northeast-1`
- Asie-Pacifique (Osaka) : `ap-northeast-3`
- Asie-Pacifique (Singapour) : `ap-southeast-1`
- Asie-Pacifique (Jakarta) : `ap-southeast-3`
- Asie-Pacifique (Melbourne) : `ap-southeast-4`
- Asie-Pacifique (Malaisie) : `ap-southeast-5`
- Israël (Tel Aviv) : `il-central-1`

- Les bases de données globales Neptune ne prennent pas en charge les types `db.t3.medium` d'`db.t4g.medium`instance.
- Les bases de données globales Neptune ne prennent pas en charge l'autoscaling pour les clusters de bases de données secondaires.
- Vous ne pouvez pas appliquer un groupe de paramètres personnalisés au cluster de bases de données globale pendant que vous effectuez une mise à niveau majeure de la version de cette base de données globale. À la place, créez vos groupes de paramètres personnalisés dans chaque région du cluster global, puis appliquez-les manuellement aux clusters régionaux après la mise à niveau.
- Vous ne pouvez pas arrêter ni démarrer les clusters de bases de données dans une base de données globale individuellement.
- Les instances de réplication en lecture d'un cluster de base de données secondaire peuvent redémarrer dans certaines circonstances, notamment lors des mises à niveau planifiées pendant votre période de maintenance. Si l'instance d'enregistreur du cluster de bases de données principal redémarre ou bascule, toutes les instances des régions secondaires redémarrent également. Le cluster secondaire n'est pas disponible tant que tous les réplicas ne sont pas de nouveau synchronisés avec l'instance d'enregistreur du cluster de bases de données principal.

Configuration d'une base de données mondiale Amazon Neptune

Vous pouvez créer une base de données Neptune globale de l'une des manières suivantes :

- Créez une base de données globale avec de nouveaux clusters de bases de données et de nouvelles instances de base de données.
- Créez une base de données globale en utilisant un cluster de bases de données Neptune existant comme cluster principal.

Rubriques

- Exigences de configuration d'une base de données globale dans Amazon Neptune
- Utilisation du AWS CLI pour créer une base de données globale dans Amazon Neptune
- Conversion d'un cluster de bases de données existant en base de données globale
- Ajout de régions de base de données globales secondaires à une région principale dans Amazon Neptune
- Connexion à une base de données Neptune globale

Exigences de configuration d'une base de données globale dans Amazon Neptune

Une base de données mondiale Neptune comprend au moins deux bases de données. Régions AWS La Région AWS principale contient un cluster de bases de données Neptune qui dispose d'une seule instance d'enregistreur. Une à cinq Régions AWS secondaires contiennent chacune un cluster de bases de données Neptune en lecture seule entièrement composé d'instances de réplica en lecture. Au moins un secondaire Région AWS est requis.

Les exigences spécifiques suivantes s'appliquent aux clusters de bases de données Neptune qui composent une base de données globale :

- Exigences relatives aux classes d'instances de base de données : une base de données globale nécessite des classes d'instances de base de données `r5` ou `r6g` optimisées pour les charges de travail gourmandes en mémoire, telles qu'un type d'instance `db.r5.large`.
- Région AWS exigences — Une base de données globale a besoin d'un cluster de base de données Neptune principal dans un Région AWS cluster de base de données Neptune et d'au moins un cluster de base de données Neptune secondaire dans une région différente. Vous pouvez créer jusqu'à cinq clusters de bases de données Neptune secondaires en lecture seule, chacun dans une région différente. En d'autres termes, deux clusters de bases de données Neptune d'une base de données globale Neptune ne peuvent pas se trouver dans la même Région AWS.

- Exigences relatives à la version du moteur : la version de moteur Neptune utilisée par tous les clusters de bases de données de la base de données globale doit être identique et doit être supérieure ou égale à. `1.2.0.0` Si vous ne spécifiez pas la version du moteur lors de la création d'une base de données globale, d'un cluster ou d'une instance, la version du moteur la plus récente est utilisée.

> **Important**
>
> Bien que les groupes de paramètres du cluster de bases de données puissent être configurés indépendamment pour chaque cluster d'une base de données globale, il est préférable de veiller à la cohérence des paramètres entre les clusters afin d'éviter des changements de comportement inattendus au cas où un cluster secondaire serait promu au statut principal. Par exemple, utilisez les mêmes paramètres pour les index d'objets, les flux, etc. dans tous les clusters de bases de données.

Utilisation du AWS CLI pour créer une base de données globale dans Amazon Neptune

> **Note**
>
> Les exemples présentés dans cette section suivent la convention UNIX qui consiste à utiliser une barre oblique inverse (\) comme caractère d'extension de ligne. Pour Windows, remplacez la barre oblique inverse par un accent circonflexe (^).

Pour créer une base de données globale à l'aide du AWS CLI

1. Commencez par créer une base de données globale vide à l'aide de la `create-global-cluster` AWS CLI commande (qui enveloppe l'CreateGlobalClusterAPI). Spécifiez le nom de la Région AWS que vous souhaitez utiliser comme région principale, définissez Neptune comme moteur de base de données et, si vous le souhaitez, spécifiez la version du moteur que vous souhaitez utiliser (il doit s'agir de la version 1.2.0.0 ou supérieure) :

```
aws neptune create-global-cluster
  --region (primary region, such as us-east-1) \
  --global-cluster-identifier (ID for the global database) \
```

```
  --engine neptune \
  --engine-version (engine version; this is optional)
```

2. Quelques minutes peuvent s'écouler avant que la base de données globale ne soit disponible. Par conséquent, avant de passer à l'étape suivante, utilisez la commande d'interface `describe-global-clusters` (qui enveloppe l'API DescribeGlobalClusters) pour vérifier que la base de données globale est disponible :

```
aws neptune describe-global-clusters \
  --region (primary region) \
  --global-cluster-identifier (global database ID)
```

3. Une fois que la base de données globale Neptune est disponible, vous pouvez créer un cluster de bases de données Neptune qui sera son cluster principal :

```
aws neptune create-db-cluster \
  --region (primary region) \
  --db-cluster-identifier (ID for the primary DB cluster) \
  --engine neptune \
  --engine-version (engine version; must be >= 1.2.0.0) \
  --global-cluster-identifier (global database ID)
```

4. Utilisez la `describe-db-clusters` AWS CLI commande pour confirmer que le nouveau cluster de base de données est prêt pour que vous puissiez ajouter son instance de base de données principale :

```
aws neptune describe-db-clusters \
  --region (primary region) \
  --db-cluster-identifier (primary DB cluster ID)
```

   Lorsque la réponse affiche `"Status": "available"`, passez à l'étape suivante.

5. Créez l'instance de base de données principale pour le cluster principal à l'aide de la `create-db-instance` AWS CLI commande. Vous devez utiliser l'un des types d'instance `r5` ou `r6g` optimisés pour la mémoire, tels que `db.r5.large`.

```
aws neptune create-db-instance \
  --region (primary region) \
  --db-cluster-identifier (primary cluster ID) \
  --db-instance-class (instance class) \
  --db-instance-identifier (ID for the DB instance) \
```

```
  --engine neptune \
  --engine-version (optional: engine version)
```

> **Note**
>
> Si vous prévoyez d'ajouter des données au nouveau cluster de bases de données principal à l'aide du chargeur en bloc Neptune, procédez avant d'ajouter des régions secondaires. Cette approche est plus rapide et plus rentable que d'effectuer un chargement en bloc une fois la base de données globale entièrement configurée.

Ajoutez maintenant une ou plusieurs régions secondaires à la nouvelle base de données globale, comme décrit dans la section Ajouter une région secondaire à l'aide du AWS CLI.

Conversion d'un cluster de bases de données existant en base de données globale

Pour transformer un cluster de bases de données existant en base de données globale, utilisez la `create-global-cluster` AWS CLI commande pour créer une nouvelle base de données globale dans le même emplacement Région AWS que le cluster de bases de données existant, et définissez son `--source-db-cluster-identifier` paramètre sur le nom de ressource Amazon (ARN) du cluster existant qui s'y trouve :

```
aws neptune create-global-cluster \
  --region (region where the existing cluster is located) \
  --global-cluster-identifier (provide an ID for the new global database) \
  --source-db-cluster-identifier (the ARN of the existing DB cluster) \
  --engine neptune \
  --engine-version (engine version; this is optional)
```

Ajoutez maintenant une ou plusieurs régions secondaires à la nouvelle base de données globale, comme décrit dans la section Ajouter une région secondaire à l'aide du AWS CLI.

Utilisation d'un cluster de bases de données restauré à partir d'un instantané comme cluster principal

Vous pouvez convertir un cluster de bases de données restauré en base de données globale Neptune à partir d'un instantané. Une fois la restauration terminée, convertissez le cluster de bases

de données créé en cluster principal d'une nouvelle base de données globale, comme décrit ci-dessus.

Ajout de régions de base de données globales secondaires à une région principale dans Amazon Neptune

Une base de données globale Neptune a besoin d'au moins un cluster de base de données Neptune secondaire dans un cluster de base de données Région AWS différent du cluster de base de données principal. Vous pouvez attacher jusqu'à cinq clusters de bases de données secondaires au cluster de bases de données principal.

Chaque cluster de bases de données secondaire que vous ajoutez réduit d'une unité le nombre maximum d'instances de réplicas en lecture que vous pouvez avoir sur le cluster principal. Par exemple, s'il existe quatre clusters secondaires, le nombre maximum d'instances de réplica en lecture que vous pouvez avoir sur le cluster principal est de `15 - 4 = 11`. Par exemple, si vous avez 14 instances de lecteur dans le cluster de bases de données principal et un seul cluster secondaire, vous ne pouvez pas ajouter de cluster secondaire.

Utilisation du AWS CLI pour ajouter une région secondaire à une base de données globale dans Neptune

Pour ajouter une base de données secondaire Région AWS à une base de données globale Neptune à l'aide du AWS CLI

1. Utilisez la `create-db-cluster` AWS CLI commande pour créer un nouveau cluster de base de données dans une région différente de celle de votre cluster principal et définissez son `--global-cluster-identifier` paramètre pour spécifier l'ID de la base de données globale :

   ```
   aws neptune create-db-cluster \
     --region (the secondary region) \
     --db-cluster-identifier (ID for the new secondary DB cluster) \
     --global-cluster-identifier (global database ID)
     --engine neptune \
     --engine-version (optional: engine version)
   ```

2. Utilisez la `describe-db-clusters` AWS CLI commande pour confirmer que le nouveau cluster de base de données est prêt pour que vous puissiez ajouter son instance de base de données principale :

```
aws neptune describe-db-clusters \
  --region (primary region) \
  --db-cluster-identifier (primary DB cluster ID)
```

Lorsque la réponse affiche `"Status": "available"`, passez à l'étape suivante.

3. Créez l'instance de base de données principale pour le cluster principal à l'aide de la `create-db-instance` AWS CLI commande, en utilisant un type d'instance dans la classe d'`r6`instance `r5` or :

```
aws neptune create-db-instance \
  --region (secondary region) \
  --db-cluster-identifier (secondary cluster ID) \
  --db-instance-class (instance class) \
  --db-instance-identifier (ID for the DB instance) \
  --engine neptune \
  --engine-version (optional: engine version)
```

> **Note**
>
> Si vous n'avez pas l'intention de traiter un grand nombre de demandes de lecture dans la région secondaire et que vous souhaitez principalement y conserver vos données de manière fiable, vous pouvez créer un cluster de bases de données secondaire sans instances de base de données. Vous réalisez ainsi des économies, car vous ne payez alors que les frais correspondant au stockage du cluster secondaire, que Neptune synchronisera avec le stockage du cluster de bases de données principal.

Connexion à une base de données Neptune globale

Vous ne vous connectez pas de la même façon à une base de données Neptune globale selon que vous devez y écrire ou y lire des données :

- Pour les demandes ou requêtes en lecture seule, vous vous connectez au point de terminaison de lecteur du cluster Neptune dans votre Région AWS.

- Pour exécuter des requêtes de mutation, connectez-vous au point de terminaison du cluster de base de données principal, qui peut se trouver dans une autre application Région AWS que celle de votre application.

Gestion d'une base de données Amazon Neptune globale

À l'exception du processus de basculement planifié géré, vous effectuez la plupart des opérations de gestion sur les clusters individuels constituant une base de données Neptune globale. Le processus de basculement planifié géré est disponible uniquement pour les bases de données Neptune globales, pas pour les clusters de bases de données Neptune individuels. Pour en savoir plus, veuillez consulter la section Basculement planifié géré pour les bases de données Neptune globales.

Pour restaurer une base de données Neptune globale suite à une panne non planifiée dans sa région principale, consultez Detach-and-promote une base de données globale Neptune en cas de panne imprévue.

Vous pouvez configurer les groupes de paramètres de cluster de bases de données indépendamment pour chaque cluster Neptune d'une base de données globale, mais il est préférable d'utiliser des paramètres cohérents entre tous les clusters afin d'éviter tout comportement inattendu en cas de promotion d'un cluster secondaire en tant que cluster principal. Par exemple, utilisez les mêmes paramètres pour les index d'objets, les flux, etc. dans tous les clusters de bases de données.

Dissociation d'un cluster d'une base de données Neptune globale

Il existe plusieurs raisons pour lesquelles vous pouvez souhaiter supprimer un cluster de bases de données d'une base de données globale. Par exemple :

- Si le cluster principal se dégrade ou devient isolé, vous pouvez le supprimer de la base de données globale pour qu'il devienne un cluster provisionné autonome qui permettra de créer une autre base de données globale (voir Detach-and-promote une base de données globale Neptune en cas de panne imprévue).
- Pour supprimer une base de données globale, vous devez d'abord supprimer (dissocier) tous les clusters associés en vous occupant du cluster principal en dernier (voir Suppression d'une base de données Neptune globale).

Vous pouvez utiliser la commande remove-from-global-clusterCLI (qui enveloppe l'RemoveFromGlobalClusterAPI) pour détacher un cluster de base de données Neptune d'une base de données globale :

```
aws neptune remove-from-global-cluster \
  --region (region of the cluster to remove) \
  --global-cluster-identifier (global database ID) \
  --db-cluster-identifier (ARN of the cluster to remove)
```

Le cluster de bases de données dissocié devient alors un cluster de bases de données autonome.

Suppression d'une base de données Neptune globale

Vous ne pouvez pas supprimer une base de données globale et les clusters associés en une seule étape. Au lieu de cela, vous devez supprimer ses composants un par un :

1. Dissociez tous les clusters de bases de données secondaires de la base de données globale, comme décrit dans Suppression d'un cluster. Si vous le souhaitez, vous pouvez maintenant les supprimer individuellement.
2. Dissociez le cluster de bases de données principal de la base de données globale.
3. Supprimez toutes les instances de base de données de réplica en lecture du cluster principal.
4. Supprimez l'instance de base de données principale (enregistreur) du cluster principal. Si vous effectuez cette opération sur la console, le cluster de bases de données est également supprimé.
5. Supprimez la base de données globale elle-même. Pour ce faire AWS CLI, utilisez la commande delete-global-clusterCLI (qui enveloppe l'DeleteGlobalClusterAPI), comme suit :

   ```
   aws neptune delete-global-cluster \
     --region (region of the DB cluster to delete) \
     --global-cluster-identifier (global database ID)
   ```

Modification d'une base de données globale Neptune

Les groupes de paramètres du cluster de bases de données peuvent être configurés indépendamment pour chaque cluster d'une base de données Neptune globale. Toutefois, il est préférable de veiller à la cohérence des paramètres entre les clusters afin d'éviter des changements de comportement inattendus au cas où un cluster secondaire serait promu au statut principal.

Vous pouvez modifier les paramètres de la base de données globale elle-même à l'aide de la commande modify-global-clusterCLI (qui enveloppe l'ModifyGlobalClusterAPI). Par exemple, vous pouvez modifier l'identifiant de la base de données globale tout en désactivant la protection contre la suppression comme suit :

```
aws neptune modify-global-cluster \
  --region (region of the DB cluster to modify) \
  --global-cluster-identifier (current global database ID) \
  --new-global-cluster-identifier (new global database ID to assign) \
  --no-deletion-protection
```

Exécution d'une reprise après sinistre | Amazon Neptune

Une base de données Neptune globale fournit des fonctionnalités de basculement plus complètes qu'un cluster de bases de données Neptune autonome. En utilisant une base de données globale, vous pouvez planifier une reprise après sinistre et l'effectuer rapidement. La reprise après sinistre est généralement évaluée à l'aide de de l'objectif de délai de reprise (RTO) et de l'objectif de point de reprise (RPO) :

- Objectif de délai de reprise (RTO) : rapidité avec laquelle un système revient à un état de fonctionnement normal après un sinistre. En d'autres termes, le RTO mesure la durée d'indisponibilité. Pour une base de données Neptune globale, le RTO peut être de l'ordre de quelques minutes.
- Objectif de point de reprise (RPO) : durée pendant laquelle les données sont perdues. Pour une base de données Neptune globale, le RPO est généralement mesuré en secondes (voir Basculement planifié géré pour les bases de données Neptune globales).

Une base de données Neptune globale permet deux approches différentes de basculement :

- Detach-and-promote (restauration non planifiée manuelle) — Pour effectuer une restauration après une panne imprévue ou pour effectuer des tests de reprise après sinistre (test DR), effectuez une opération entre régions detach-and-promote sur l'un des clusters de base de données secondaires de la base de données globale. Pour ce processus, le RTO manuel dépend de la rapidité avec laquelle vous pouvez effectuer les tâches répertoriées dans Dissociation et promotion. Le RPO est généralement mesuré en secondes, mais cela dépend du retard de réplication du stockage sur le réseau au moment de l'interruption.

- Basculement planifié géré : cette approche est destinée à la maintenance opérationnelle et aux autres procédures opérationnelles planifiées, telles que le déplacement du cluster principal de la base de données globale vers l'une des régions secondaires. Étant donné que cette fonction synchronise les clusters de bases de données secondaires avec le cluster principal avant toute modification, le RPO est égal à 0 (aucune donnée perdue). Consultez Basculement planifié géré pour les bases de données Neptune globales.

Detach-and-promote une base de données globale Neptune en cas de panne imprévue

Dans les très rares cas où votre base de données globale Neptune subit une panne inattendue dans sa base de données principale Région AWS, votre cluster de base de données Neptune principal et son nœud d'écriture deviennent indisponibles, et la réplication entre le cluster principal et les secondaires cesse. Pour minimiser à la fois les temps d'arrêt (RTO) et les pertes de données (RPO) qui en résultent, effectuez rapidement une analyse interrégionale detach-and-promote pour reconstruire la base de données globale.

> **Tip**
>
> Il est conseillé de comprendre ce processus avant de l'utiliser et d'avoir une stratégie en place pour agir rapidement dès les premiers signes d'un problème à l'échelle de la région.
>
> - Utilisez Amazon CloudWatch régulièrement pour suivre les temps de latence des clusters secondaires afin d'identifier la région secondaire présentant le moins de latence en cas de basculement.
> - Assurez-vous de tester cette stratégie pour vérifier que les procédures sont complètes et précises.
> - Utilisez un environnement simulé pour vous assurer que votre personnel est formé et prêt à effectuer rapidement un basculement après sinistre si cela s'avère nécessaire.

Pour basculer vers un cluster secondaire après une interruption non planifiée dans la région principale

1. Arrêtez d'émettre des requêtes de mutation et d'autres opérations d'écriture sur le cluster de bases de données principal.

2. Identifiez un cluster de base de données dans un cluster secondaire Région AWS à utiliser comme nouveau cluster de base de données principal de la base de données globale. Si la base de données globale compte deux Régions AWS secondaires (ou plus), choisissez le cluster secondaire ayant le retard le moins important.
3. Dissociez le cluster de bases de données secondaire que vous avez choisi de la base de données Neptune globale.

   La suppression d'un cluster de base de données secondaire d'une base de données globale Neptune arrête immédiatement la réplication des données du cluster principal vers ce cluster secondaire et en fait un cluster de base de données autonome doté de fonctionnalités complètes. read/write Tous les autres clusters secondaires de la base de données globale restent disponibles et peuvent accepter les appels de lecture à partir de votre application.

   Avant de recréer la base de données Neptune globale, vous devez également dissocier les autres clusters secondaires pour éviter les incohérences de données entre clusters (voir Suppression d'un cluster).
4. Reconfigurez votre application pour envoyer toutes les opérations d'écriture au cluster de bases de données Neptune autonome que vous avez choisi en tant que nouveau cluster principal, à l'aide de son nouveau point de terminaison. Si vous avez accepté les noms par défaut lors de la création de la base de données Neptune globale, vous pouvez modifier le point de terminaison en supprimant `-ro` de la chaîne du point de terminaison de cluster dans votre application.

   Par exemple, le point de terminaison du cluster secondaire `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com` devient `my-global.cluster-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com` lorsque ce cluster est dissocié de la base de données globale.

   Ce cluster de bases de données Neptune devient le cluster principal d'une nouvelle base de données Neptune globale lorsque vous commencez à y ajouter des régions lors de l'étape suivante.
5. Ajoutez un Région AWS au cluster de base de données. Lorsque vous effectuez cette opération, le processus de réplication du cluster primaire vers le cluster secondaire commence. Consultez Ajout de régions de base de données globales secondaires à une région principale dans Amazon Neptune.
6. Ajoutez-en Régions AWS d'autres si nécessaire pour recréer la topologie requise pour prendre en charge votre application.

Assurez-vous que les écritures d'application sont envoyées au cluster de bases de données Neptune approprié avant, pendant et après l'application de ces modifications. Cela évite les incohérences de données parmi les clusters de la base de données Neptune globale (ce que l'on appelle aussi « problèmes de split-brain »).

Basculement planifié géré pour les bases de données Neptune globales

Le basculement planifié géré vous permet de déplacer le cluster principal de votre base de données globale Neptune vers un autre Région AWS cluster quand vous le souhaitez. Certaines organisations préfèrent changer régulièrement l'emplacement de leur cluster principal.

> **Note**
>
> Le processus de basculement planifié géré décrit ici est conçu pour être utilisé sur une base de données Neptune globale saine. Pour se remettre d'une interruption non planifiée ou pour effectuer des tests de reprise après sinistre, suivez plutôt le processus de dissociation et de promotion de cluster.

Lors d'un basculement planifié géré, le cluster principal est basculé vers la région secondaire de votre choix tandis que la topologie de réplication existante de la base de données globale est conservée. Avant le début du processus de basculement planifié géré, la base de données globale synchronise tous les clusters secondaires avec son cluster principal. Une fois que tous les clusters sont synchronisés, le basculement planifié géré commence. Le cluster de bases de données dans la région principale devient en lecture seule, et le cluster secondaire choisi promeut l'une de ses instances en lecture seule au statut d'enregistreur complet, permettant ainsi au cluster d'endosser le rôle de cluster principal. Comme tous les clusters secondaires ont été synchronisés avec le cluster principal au début du processus, le nouveau cluster principal poursuit les opérations de la base de données globale sans perdre de données. La base de données est uniquement indisponible pendant une courte période durant laquelle le cluster principal et les clusters secondaires sélectionnés endossent leur nouveau rôle.

Pour optimiser la disponibilité des applications, effectuez le basculement pendant les heures creuses, à un moment où les écritures sur le cluster de bases de données principal sont minimales. Procédez également comme suit avant de démarrer le basculement :

- Mettez les applications hors ligne dans la mesure du possible afin de réduire les écritures sur le cluster principal.

- Vérifiez les temps de latence pour tous les clusters Neptune secondaires dans la base de données globale et choisissez le cluster secondaire présentant le moins de retard global pour qu'il devienne le cluster principal. Utilisez Amazon CloudWatch pour consulter les statistiques `NeptuneGlobalDBProgressLag` de tous les établissements secondaires. Cette métrique indique le retard d'un cluster secondaire (en millisecondes) par rapport au cluster de bases de données principal. Sa valeur est directement proportionnelle au temps nécessaire à Neptune pour terminer le basculement. En d'autres termes, plus la valeur de retard est élevée, plus le basculement sera long. Choisissez donc le cluster secondaire avec le moins de retard. Pour plus d'informations, consultez Métriques Neptune CloudWatch.

Au cours d'un basculement planifié géré, le cluster de bases de données secondaire choisi est promu à son nouveau rôle de cluster principal, mais il n'hérite pas de la configuration complète du cluster de bases de données principal. Une incompatibilité de configuration peut provoquer des problèmes de performances, des incompatibilités de charge de travail et d'autres comportements anormaux. Pour éviter de tels problèmes, corrigez les types de différences de configuration suivants entre les clusters de la base de données globale avant le basculement :

- Configurez les paramètres du nouveau cluster principal pour qu'ils correspondent à ceux du cluster principal actuel.
- Configurez les outils, les options et les alarmes de surveillance : configurez le cluster de bases de données qui sera le nouveau cluster principal avec la même capacité de journalisation, les mêmes alarmes, etc. que le cluster principal actuel.
- Configurez les intégrations avec d'autres AWS services : si votre base de données globale Neptune s'intègre AWS à des services Gestion des identités et des accès AWS tels que (IAM), Amazon S3, AWS Lambda ou assurez-vous que ceux-ci sont configurés selon les besoins pour s'intégrer au nouveau cluster de base de données principal.

Lorsque le processus de basculement est terminé et que le cluster de bases de données promu est prêt à gérer les opérations d'écriture pour la base de données globale, veillez à modifier vos applications afin qu'elles utilisent le nouveau point de terminaison du nouveau cluster principal.

Utilisation du AWS CLI pour lancer un basculement planifié géré

Utilisez la commande failover-global-clusterCLI (qui enveloppe l'FailoverGlobalClusterAPI) pour basculer sur votre base de données globale Neptune :

```
aws neptune failover-global-cluster \
```

```
  --region (the region where the primary cluster is located) \
  --global-cluster-identifier (global database ID) \
  --target-db-cluster-identifier (the ARN of the secondary DB cluster to promote)
```

Surveillance de votre base de données mondiale Amazon Neptune à l'aide de métriques CloudWatch

Neptune prend en charge les CloudWatch mesures suivantes que vous pouvez utiliser pour surveiller une base de données globale Neptune :

- **`GlobalDbDataTransferBytes`**— Le nombre d'octets de données de journalisation transférés du primaire Région AWS vers le secondaire Région AWS dans une base de données globale Neptune.
- **`GlobalDbReplicatedWriteIO`**— Nombre d' I/O opérations d'écriture répliquées depuis le volume principal Région AWS de la base de données globale vers le volume de cluster d'un volume secondaire Région AWS.

  Les calculs de facturation pour chaque cluster de bases de données dans une base de données globale Neptune utilisent la métrique `VolumeWriteIOPS` pour prendre en compte les écritures effectuées dans ce cluster. Pour le cluster de bases de données principal, les calculs de facturation utilisent `NeptuneGlobalDbReplicatedWriteIO` afin de prendre en compte la réplication entre régions vers les clusters de bases de données secondaires.
- **`GlobalDbProgressLag`** : retard du cluster secondaire en millisecondes par rapport au cluster principal pour les transactions utilisateur et système.

Métrique	Dimension	Publiée dans	Unités
`GlobalDbDataTransferBytes`	SourceRegion, DBCluster Identifiant	Secondary	Octets
`GlobalDbReplicatedWriteIO`	SourceRegion, DBCluster Identifiant	Secondary	Nombre

Métrique	Dimension	Publiée dans	Unités
`GlobalDbProgressLag`	DBClusterIdentifiant, SecondaryRegion : dans DBCluster Identifiant principal, SourceRegion : dans secondaire	Principal, secondaire	Millisecondes

Présentation des fonctionnalités Amazon Neptune

> **Note**
>
> Cette section ne traite pas de l'utilisation des langages de requête permettant d'accéder aux données d'un graphe Neptune.
> Pour en savoir plus sur la connexion d'un cluster de bases de données Neptune en cours d'exécution avec Gremlin, consultez Accès au graphe Neptune avec Gremlin.
> Pour en savoir plus sur la connexion d'un cluster de bases de données Neptune en cours d'exécution avec openCypher, consultez Accès au graphe Neptune avec openCypher.
> Pour en savoir plus sur la connexion d'un cluster de bases de données Neptune en cours d'exécution avec SPARQL, consultez Accès au graphe Neptune avec SPARQL.

Cette section présente les fonctionnalités spécifiques de Neptune, notamment :

- Conformité de Neptune avec les normes des langages de requête
- Modèle de données de graphe de Neptune
- Explication de la sémantique des transactions Neptune
- Présentation des clusters et instances Neptune
- Stockage, fiabilité et disponibilité de Neptune
- Explication des points de terminaison Neptune
- Utilisation du mode laboratoire de Neptune pour activer les fonctionnalités expérimentales
- Description du moteur DFE de Neptune
- Connectivité JDBC de Neptune
- Liste des versions du moteur Neptune et procédure de mise à jour de votre moteur

Remarques sur la conformité d'Amazon Neptune avec les normes

Amazon Neptune est conforme aux normes applicables à l'implémentation des langages de requête de graphe Gremlin et SPARQL dans la plupart des cas.

Ces sections décrivent les normes, ainsi que les domaines où Neptune étend ces normes ou diverge de celles-ci.

Rubriques

- Conformité d'Amazon Neptune avec les normes Gremlin
- Conformité d'Amazon Neptune avec les normes SPARQL
- Conformité aux spécifications OpenCypher dans Amazon Neptune

Conformité d'Amazon Neptune avec les normes Gremlin

Les sections suivantes fournissent une vue d'ensemble de l'implémentation Neptune de Gkremlin et de ses différences par rapport à l'implémentation Apache. TinkerPop

Neptune implémente certaines étapes de Gkremlin de manière native dans son moteur et utilise l'implémentation d'Apache TinkerPop Gkremlin pour en traiter d'autres (voir). Prise en charge des étapes Gremlin natives dans Amazon Neptune

> **Note**
>
> Pour obtenir des exemples concrets de ces différences implémentation dans la console Gremlin et dans Amazon Neptune, consultez la section the section called “Utilisation de Gremlin” du Quick Start.

Rubriques

- Normes applicables pour Gremlin
- Variables et paramètres dans les scripts
- TinkerPop énumérations
- Code Java
- Propriétés des éléments
- Exécution de script

- Séances
- Transactions
- ID de sommet et ID d'arête
- User-supplied Identifiants
- ID de propriété de vertex
- Cardinalité des propriétés de sommet
- Mise à jour d'une propriété de sommet
- Étiquettes
- Caractères d'échappement
- Limites Groovy
- Sérialisation
- Étapes Lambda
- Méthodes Gremlin non prises en charge
- Étapes Gremlin non prises en charge
- Fonctionnalités du graphe Gremlin dans Neptune

Normes applicables pour Gremlin

- Le langage G705 est défini par la TinkerPop documentation Apache et l' TinkerPop implémentation Apache de G705 plutôt que par une spécification formelle.
- Pour les formats numériques, G705 suit la norme IEEE 754 (IEEE 754-2019 - IEEE Standard for Arithmetic). Floating-Point Pour plus d'informations, consultez également la page Wikipedia (IEEE 754).

Variables et paramètres dans les scripts

En ce qui concerne les variables prédéfinies, l'objet de traversée g se trouve dans Pre-bound Neptune et l'`graph`objet n'est pas pris en charge.

Bien que Neptune ne prenne pas en charge les variables Gremlin ni le paramétrage dans des scripts, vous pouvez souvent rencontrer sur Internet des exemples de scripts contenant des déclarations de variables, tels que :

```
String query = "x = 1; g.V(x)";
```

```
List<Result> results = client.submit(query).all().get();
```

Il existe également de nombreux exemples qui utilisent le paramétrage (ou les liaisons) lors de la soumission de requêtes, tels que :

```
Map<String,Object> params = new HashMap<>();
params.put("x",1);
String query = "g.V(x)";
List<Result> results = client.submit(query).all().get();
```

Ces exemples de paramètres sont généralement associés à des avertissements sur les pénalités de performance possibles en cas de non-paramétrage lorsque cela est possible. Il existe de nombreux exemples de ce type TinkerPop que vous pouvez rencontrer, et ils semblent tous assez convaincants quant à la nécessité de paramétrer.

Cependant, les fonctionnalités de déclaration de variables et de paramétrage (ainsi que les avertissements) ne s'appliquent qu'au TinkerPop serveur G705 lorsqu'il utilise le. `GremlinGroovyScriptEngine` Elles ne s'appliquent pas lorsque le serveur Gremlin utilise la grammaire `gremlin-language` ANTLR de Gremlin pour analyser les requêtes. La grammaire ANTLR ne prend en charge ni les déclarations de variables ni le paramétrage. Ainsi, lorsque vous utilisez ANTLR, vous n'avez rien à craindre en cas de non-paramétrage. La grammaire ANTLR étant une composante plus récente TinkerPop, les anciens contenus que vous pouvez rencontrer sur Internet ne reflètent généralement pas cette distinction.

Neptune utilise la grammaire ANTLR dans son moteur de traitement des requêtes plutôt que le moteur `GremlinGroovyScriptEngine`. Il ne prend donc pas en charge les variables, le paramétrage ni la propriété `bindings`. Par conséquent, les problèmes potentiels liés au non-paramétrage ne s'appliquent pas dans Neptune. Avec Neptune, il est parfaitement sûr de soumettre simplement la requête telle quelle, alors que beaucoup la paramétrerait. Par conséquent, l'exemple précédent peut être simplifié sans aucune perte de performance comme suit :

```
String query = "g.V(1)";
List<Result> results = client.submit(query).all().get();
```

TinkerPop énumérations

Neptune ne prend pas en charge les noms de classe complets pour les valeurs d'énumération. Par exemple, vous devez utiliser `single` et non

`org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single` dans votre demande Groovy.

Le type d'énumération est déterminé par le type de paramètre.

Le tableau suivant indique les valeurs d'énumération autorisées et le nom TinkerPop complet correspondant.

Valeurs autorisées	Classe
`id`, `key`, `label`, `value`	org.apache.tinkerpop.gremlin.structure.T
`T.id`, `T.key`, `T.label`, `T.value`	org.apache.tinkerpop.gremlin.structure.T
`set`, `single`	org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality
`asc`, `desc`, `shuffle`	org.apache.tinkerpop.gremlin.process.traversal.Order
`Order.asc` , `Order.desc` , `Order.shuffle`	org.apache.tinkerpop.gremlin.process.traversal.Order
`global`, `local`	org.apache.tinkerpop.gremlin.process.traversal.Scope
`Scope.global` , `Scope.local`	org.apache.tinkerpop.gremlin.process.traversal.Scope
`all`, `first`, `last`, `mixed`	org.apache.tinkerpop.gremlin.process.traversal.Pop
`normSack`	org.apache.tinkerpop.gremlin.process.traversal.SackFunctions.Barrier
`addAll`, `and`, `assign`, `div`, `max`, `min`, `minus`, `mult`, `or`, `sum`, `sumLong`	org.apache.tinkerpop.gremlin.process.traversal.Operator
`keys`, `values`	org.apache.tinkerpop.gremlin.structure.Column

`BOTH`, `IN`, `OUT`	org.apache.tinkerpop.gremlin.structure.Direction
`any`, `none`	org.apache.tinkerpop.gremlin.process.traversal.step. TraversalOptionParent.Pick

Code Java

Neptune ne prend pas en charge les appels de méthodes définies par des appels Java ou des appels de bibliothèque Java arbitraires autres que les API Gremlin prises en charge. Par exemple, `java.lang.*`, `Date()` et `g.V().tryNext().orElseGet()` ne sont pas autorisés.

Propriétés des éléments

Neptune ne prend pas en charge l'`materializeProperties`indicateur introduit dans la TinkerPop version 3.7.0 pour renvoyer les propriétés des éléments. Par conséquent, Neptune ne renverra toujours que les sommets ou les arêtes en tant que références avec uniquement leur et. `id` `label`

Exécution de script

Toutes les requêtes doivent commencer par g qui est l'objet de traversée.

Dans les soumissions de requête de type Chaîne, plusieurs traversées peuvent être émises, séparées par un point-virgule (`;`) ou un caractère de saut de ligne (`\n`). Pour être exécutée, chaque instruction autre que la dernière doit se terminer par une étape `.iterate()`. Seules les données de la traversée finale sont renvoyées. Notez que cela ne s'applique pas aux soumissions de ByteCode requêtes GLV.

Séances

Les sessions dans Neptune sont limitées à seulement 10 minutes. Voir Sessions basées sur des scripts Gremlin et la référence de TinkerPop session pour plus d'informations.

Transactions

Neptune ouvre une nouvelle transaction au début de chaque traversée Gremlin et ferme la transaction lors de la réussite complète de la traversée. La transaction est annulée lorsqu'il y a une erreur.

Plusieurs instructions séparées par un point-virgule (`;`) ou un caractère de nouvelle ligne (`\n`) sont incluses dans une seule transaction. Chaque instruction autre que la dernière doit se terminer par une étape `next()` à exécuter. Seules les données de la traversée finale sont renvoyées.

La logique de transaction manuelle utilisant `tx.commit()` et `tx.rollback()` n'est pas prise en charge.

> **Important**
>
> Ceci s'applique uniquement aux méthodes dans lesquelles vous envoyez la requête Gremlin en tant que chaîne de texte (voir Transactions Gremlin).

ID de sommet et ID d'arête

Les ID de sommet et d'arête de Neptune doivent être de type `String`. Ces chaînes d'ID prennent en charge les caractères Unicode et ne peuvent pas dépasser 55 Mo.

User-supplied Les identifiants sont pris en charge, mais ils sont facultatifs dans le cadre d'une utilisation normale. Si vous ne fournissez pas d'ID lorsque vous ajoutez un sommet ou une arête, Neptune génère un UUID et le convertit en chaîne, sous une forme similaire à celle-ci : `"48af8178-50ce-971a-fc41-8c9a954cea62"`. Ces UUID ne sont pas conformes à la norme RFC. Par conséquent, si vous avez besoin d'UUID standard, vous devrez les générer en externe et les fournir lorsque vous ajouterez des sommets ou des arêtes.

> **Note**
>
> La commande Neptune `Load` nécessite que vous fournissiez les ID à l'aide du champ ~ id dans le format CSV de Neptune.

User-supplied Identifiants

User-supplied Les pièces d'identité sont autorisées dans Neptune Gremlin sous réserve des conditions suivantes.

- Les ID fournis sont facultatifs.
- Seuls les vertex et les edges sont pris en charge.

- Seul le type `String` est pris en charge.

Pour créer un nouveau vertex avec un ID personnalisé, utilisez l'étape `property` avec le mot-clé `id` : `g.addV().property(id, 'customid')`.

> **Note**
>
> Ne placez pas de guillemets autour du mot-clé `id`. Il fait référence à `T.id`.

Tous les ID vertex et ID edge doivent être uniques. Cependant, Neptune permet d'avoir le même ID pour un sommet et une arête.

Si vous essayez de créer un nouveau vertex à l'aide de `g.addV()` et qu'il existe déjà un vertex ayant cet ID, l'opération échoue. L'exception à cette règle, c'est que si vous précisez une nouvelle étiquette pour le vertex, l'opération réussit, mais elle ajoute la nouvelle étiquette et toute propriété supplémentaire précisée au sommet existant. Rien n'est remplacé. Un nouveau vertex n'est pas créé. L'ID de sommet ne change pas et reste unique.

Par exemple, les commandes suivantes de la console Gremlin aboutissent :

```
gremlin> g.addV('label1').property(id, 'customid')
gremlin> g.addV('label2').property(id, 'customid')
gremlin> g.V('customid').label()
==>label1::label2
```

ID de propriété de vertex

Les ID de propriété de vertex sont générés automatiquement et peuvent s'afficher comme nombres positifs ou négatifs lors des requêtes.

Cardinalité des propriétés de sommet

Neptune prend en charge la cardinalité définie et la cardinalité unique. Si elle n'est pas spécifiée, la cardinalité définie est sélectionnée. Cela signifie que si vous définissez une valeur de propriété, une nouvelle valeur est ajoutée à la propriété, mais uniquement si elle n'apparaît pas déjà dans l'ensemble de valeurs. Il s'agit de la valeur d'énumération Gremlin Set.

`List` n'est pas pris en charge. Pour plus d'informations sur la cardinalité des propriétés, consultez la rubrique Vertex dans le Gkremlin. JavaDoc

Mise à jour d'une propriété de sommet

Pour mettre à jour une valeur de propriété sans ajouter une valeur à l'ensemble des valeurs, spécifiez la cardinalité `single` lors de l'étape `property`.

```
g.V('exampleid01').property(single, 'age', 25)
```

Cela supprime toutes les valeurs existantes de la propriété.

Étiquettes

Neptune prend en charge plusieurs étiquettes pour un sommet. Lorsque vous créez une étiquette, vous pouvez spécifier plusieurs étiquettes en les séparant par `::`. Par exemple, `g.addV("Label1::Label2::Label3")` ajoute un vertex, avec trois étiquettes différentes. L'étape `hasLabel` associe ce sommet à l'une de ces trois étiquettes : `hasLabel("Label1")` `hasLabel("Label2")` et `hasLabel("Label3")`.

> **Important**
>
> Le délimiteur `::` est réservé à cet usage uniquement. Vous ne pouvez pas spécifier plusieurs étiquettes dans l'étape `hasLabel`. Par exemple, `hasLabel("Label1::Label2")` ne correspond à rien.

Caractères d'échappement

Neptune résout tous les caractères d'échappement comme décrit dans la section Escaping Special Characters (Échappement des caractères spéciaux) de la documentation du langage Apache Groovy.

Limites Groovy

Neptune ne prend pas en charge les commandes Groovy qui ne commencent pas par `g`. Cela inclut les calculs (par exemple, `1+1`), les appels système (par exemple, `System.nanoTime()`) et les définitions de variable (par exemple, `1+1`).

> **Important**
>
> Neptune ne prend pas en charge les noms de classe complets. Par exemple, vous devez utiliser `single` et non

`org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single` dans votre demande Groovy.

Sérialisation

Neptune prend en charge les sérialisations suivantes en fonction du type MIME demandé.

Neptune expose tous les sérialiseurs qui le TinkerPop font, avec le support des différentes versions et configurations de GraphSon et. GraphBinary Bien que de nombreuses options soient présentes, les conseils à suivre sont simples :

- Si vous utilisez des TinkerPop pilotes Apache, préférez le pilote par défaut sans en spécifier un explicitement. À moins que vous n'ayez une raison très précise de le faire, vous n'avez probablement pas besoin de spécifier le sérialiseur lors de l'initialisation de votre pilote. En général, la valeur par défaut utilisée par les pilotes est`application/vnd.graphbinary-v1.0`.
- Si vous vous connectez à Neptune via HTTP, privilégiez l'utilisation de `application/vnd.gremlin-v3.0+json;types=false` car les types intégrés dans la version alternative de GraphSon 3 compliquent l'utilisation.
- Le n'`application/vnd.graphbinary-v1.0-string`est généralement utile que lorsqu'il est utilisé conjointement avec la console G705, car il convertit tous les résultats en une représentation sous forme de chaîne pour un affichage simple.
- Les autres formats restent présents pour des raisons héritées du passé et ne doivent généralement pas être utilisés avec des pilotes sans raison claire.

Type MIME	Sérialisation	Configuration
`application/vnd.gremlin-v1.0+json`	`GraphSONMessageSerializerV1`	`ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV1]`
`application/vnd.gremlin-v1.0+json;types=false`	`GraphSONUntypedMessageSerializerV1`	`ioRegistries: [org.apache.tinkerpop.gremlin.tinker`

		graph.structure.TinkerIoRegistryV1]
application/vnd.gremlin-v2.0+json	GraphSONMessageSerializerV2	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV2]
application/vnd.gremlin-v2.0+json;types=false	GraphSONUntypedMessageSerializerV2	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV2]
application/vnd.gremlin-v3.0+json	GraphSONMessageSerializerV3	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV3]
application/vnd.gremlin-v3.0+json;types=false	GraphSONUntypedMessageSerializerV3	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV3]
application/json	GraphSONUntypedMessageSerializerV3	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV1]
application/vnd.graphbinary-v1.0	GraphBinaryMessageSerializerV1	

`application/vnd.graphbinary-v1.0-stringd`	`GraphBinaryMessageSerializerV1`	`serializeResultToString: true`
`application/vnd.gremlin-v1.0+json`	`GraphSONMessageSerializerGremlinV1`	`ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV1]`
`application/vnd.gremlin-v2.0+json`	`GraphSONMessageSerializerV2` (fonctionne uniquement avec WebSockets)	`ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV2]`
`application/vnd.gremlin-v3.0+json`	`GraphSONMessageSerializerV3`	
`application/json`	`GraphSONMessageSerializerV3`	`ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV3]`
`application/vnd.graphbinary-v1.0`	`GraphBinaryMessageSerializerV1`	

> **Note**
>
> Le tableau du sérialiseur présenté ici fait référence à la dénomination à partir de la version 3.7.0. TinkerPop Si vous souhaitez en savoir plus sur cette modification, consultez la documentation de TinkerPop mise à niveau. La prise en charge de la sérialisation de Gryo est devenue obsolète dans la version 3.4.3 et a été officiellement supprimée dans la version 3.6.0. Si vous utilisez explicitement Gryo ou si vous utilisez une version du pilote qui l'utilise par défaut, vous devez passer à votre pilote GraphBinary ou le mettre à niveau.

Étapes Lambda

Neptune ne prend pas en charge les étapes Lambda.

Méthodes Gremlin non prises en charge

Neptune ne prend pas en charge les méthodes Gremlin suivantes :

- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.progra`
- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.sideEf`
- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.from(o`
- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.to(org`

Par exemple, la traversée suivante n'est pas autorisée : `g.V().addE('something').from(__.V().next()).to(__.V().next())`.

> **Important**
>
> Ceci s'applique uniquement aux méthodes dans lesquelles vous envoyez la requête Gremlin en tant que chaîne de texte.

Étapes Gremlin non prises en charge

Neptune ne prend pas en charge les étapes Gremlin suivantes :

- L'étape Gremlin io () n'est que partiellement prise en charge dans Neptune. Elle peut être utilisée dans un contexte de lecture, par exemple `g.io(`*`(url)`*`).read()`, mais pas pour écrire.

Fonctionnalités du graphe Gremlin dans Neptune

L'implémentation Neptune de Gremlin n'expose pas l'objet `graph`. Les tableaux suivants répertorient les fonctionnalités Gremlin et indiquent si Neptune les prend en charge ou non.

`Support de Neptune pour les fonctionnalités graphiques`

Les fonctionnalités de graphe Neptune sont les mêmes que celles qui seraient renvoyées par la commande `graph.features()`.

Fonctionnalité de graphe	Activé ?
`Transactions`	vrai
`ThreadedTransactions`	false
`Computer`	false
`Persistence`	true
`ConcurrentAccess`	true

Prise en charge Neptune des fonctionnalités de variable

Fonctionnalité de variable	Activé ?
`Variables`	false
`SerializableValues`	false
`UniformListValues`	false
`BooleanArrayValues`	false
`DoubleArrayValues`	false
`IntegerArrayValues`	false
`StringArrayValues`	false
`BooleanValues`	false
`ByteValues`	false
`DoubleValues`	false
`FloatValues`	false
`IntegerValues`	false
`LongValues`	false

MapValues	false
MixedListValues	false
StringValues	false
ByteArrayValues	false
FloatArrayValues	false
LongArrayValues	false

Prise en charge Neptune des fonctionnalités de sommet

Fonctionnalité de sommet	Activé ?
MetaProperties	false
DuplicateMultiProperties	false
AddVertices	true
RemoveVertices	true
MultiProperties	true
UserSuppliedIds	true
AddProperty	true
RemoveProperty	vrai
NumericIds	false
StringIds	vrai
UuidIds	false
CustomIds	false
AnyIds	false

Prise en charge Neptune des fonctionnalités de propriété de sommet

Fonctionnalité de propriété de sommet	Activé ?
`UserSuppliedIds`	false
`AddProperty`	true
`RemoveProperty`	true
`NumericIds`	true
`StringIds`	vrai
`UuidIds`	false
`CustomIds`	false
`AnyIds`	false
`Properties`	vrai
`SerializableValues`	false
UniformListValues	false
`BooleanArrayValues`	false
`DoubleArrayValues`	false
`IntegerArrayValues`	false
`StringArrayValues`	false
`BooleanValues`	true
`ByteValues`	true
`DoubleValues`	true
`FloatValues`	true
`IntegerValues`	true

LongValues	vrai
MapValues	false
MixedListValues	false
StringValues	vrai
ByteArrayValues	false
FloatArrayValues	false
LongArrayValues	false

Prise en charge Neptune des fonctionnalités d'arête

Fonctionnalité d'arête	Activé ?
AddEdges	true
RemoveEdges	true
UserSuppliedIds	true
AddProperty	true
RemoveProperty	vrai
NumericIds	false
StringIds	vrai
UuidIds	false
CustomIds	false
AnyIds	false

Prise en charge Neptune des fonctionnalités de propriété d'arête

Fonctionnalité de propriété d'arête	Activé ?
`Properties`	vrai
`SerializableValues`	false
`UniformListValues`	false
`BooleanArrayValues`	false
`DoubleArrayValues`	false
`IntegerArrayValues`	false
`StringArrayValues`	false
`BooleanValues`	true
`ByteValues`	true
`DoubleValues`	true
`FloatValues`	true
`IntegerValues`	true
`LongValues`	vrai
`MapValues`	false
`MixedListValues`	false
`StringValues`	vrai
`ByteArrayValues`	false
`FloatArrayValues`	false
`LongArrayValues`	false

Conformité d'Amazon Neptune avec les normes SPARQL

Après avoir répertorié les normes SPARQL applicables, les sections suivantes fournissent des détails spécifiques sur la manière dont l'implémentation SPARQL de Neptune étend ces normes ou diverge de celles-ci.

Rubriques

- Normes applicables pour SPARQL
- Préfixes d'espace de noms par défaut dans Neptune SPARQL
- Graphe par défaut SPARQL et graphes nommés
- Fonctions de constructeur SPARQL XPath prises en charge par Neptune
- IRI de base par défaut pour les requêtes et les mises à jour
- Valeurs xsd:dateTime dans Neptune
- Traitement des valeurs spéciales à virgule flottante par Neptune
- Limite des valeurs de Neptune Arbitrary-Length
- Neptune étend la comparaison Equals dans SPARQL
- Gestion des Out-of-Range littéraux dans Neptune SPARQL

Amazon Neptune est conforme avec les normes suivantes lors de l'implémentation du langage de requête de graphe SPARQL.

Normes applicables pour SPARQL

- SPARQL est défini par la recommandation du W3C SPARQL 1.1 Query Language du 21 mars 2013.
- Le protocole de mise à jour SPARQL et le langage de requête sont définis par la spécification W3C SPARQL 1.1 Update.
- Pour les formats numériques, SPARQL suit le langage de définition de schéma XML (XSD) 1.1 du W3C, partie 2 : spécification des types de données, qui est conforme à la spécification IEEE 754 (IEEE 754-2019 - IEEE Standard for Arithmetic). Floating-Point Pour plus d'informations, consultez également la page Wikipedia (IEEE 754). Toutefois, les fonctionnalités introduites après la version `IEEE 754-1985` ne sont pas incluses dans la spécification.

Préfixes d'espace de noms par défaut dans Neptune SPARQL

Neptune définit les préfixes suivants par défaut à utiliser dans les requêtes SPARQL. Pour plus d'informations, consultez Noms préfixés dans la spécification SPARQL.

- `rdf` – `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
- `rdfs` – `http://www.w3.org/2000/01/rdf-schema#`
- `owl` – `http://www.w3.org/2002/07/owl#`
- `xsd` – `http://www.w3.org/2001/XMLSchema#`

Graphe par défaut SPARQL et graphes nommés

Amazon Neptune associe chaque triplet à un graphe nommé. Le graphe par défaut est défini comme l'union de tous les graphes nommés.

Graphe par défaut pour les requêtes

Si vous envoyez une requête SPARQL sans spécifier explicitement un graphe via le mot-clé `GRAPH` ou des constructions telles que `FROM NAMED`, Neptune prend toujours en compte tous les triplets dans votre instance de base de données. Par exemple, la requête suivante renvoie tous les triplets à partir d'un point de terminaison Neptune SPARQL :

```
SELECT * WHERE { ?s ?p ?o }
```

Les triplets qui apparaissent dans plusieurs graphes ne sont renvoyés qu'une seule fois.

Pour plus d'informations sur la spécification du graphe par défaut, consultez la section RDF Dataset dans la spécification de langage de requête SPARQL 1.1.

Spécification du graphe nommé pour le chargement, les insertions ou les mises à jour

Si vous ne spécifiez pas un graphe nommé lors du chargement, de l'insertion ou de la mise à jour de triplets, Neptune utilise le graphe nommé de secours défini par l'URI `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Lorsque vous envoyez une demande `Load` Neptune à l'aide d'un format basé sur les triplets, vous pouvez spécifier le graphe nommé à utiliser pour tous les triplets à l'aide du paramètre `parserConfiguration: namedGraphUri`. Pour plus d'informations sur la syntaxe de la commande `Load`, consultez the section called “Commande Loader”.

> **Important**
>
> Si vous n'utilisez pas ce paramètre, et que vous ne spécifiez pas un graphe nommé, l'URI de secours est utilisé : `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Ce graphe nommé de secours est également utilisé si vous chargez des triplets via `SPARQL UPDATE` sans fournir explicitement une cible de graphe nommé.

Vous pouvez utiliser le format basé sur des carrés N-Quads pour spécifier un graphe nommé pour chaque triple de la base de données.

> **Note**
>
> L'utilisation vous N-Quads permet de laisser le graphique nommé vide. Dans ce cas, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph` est utilisé. Vous pouvez remplacer le graphe nommé par défaut pour N-Quads utiliser l'option de configuration de l'`namedGraphUri`analyseur.

Fonctions de constructeur SPARQL XPath prises en charge par Neptune

La norme SPARQL permet aux moteurs SPARQL de prendre en charge un ensemble extensible de fonctions de constructeur XPath. Neptune prend actuellement en charge les fonctions de constructeur suivantes, dont le préfixe `xsd` est défini comme `http://www.w3.org/2001/XMLSchema#` :

- `xsd:boolean`
- `xsd:integer`
- `xsd:double`
- `xsd:float`
- `xsd:decimal`
- `xsd:long`
- `xsd:unsignedLong`

IRI de base par défaut pour les requêtes et les mises à jour

Comme un cluster Neptune possède plusieurs points de terminaison différents, l'utilisation de l'URL de demande d'une requête ou d'une mise à jour comme IRI de base peut entraîner des résultats inattendus lors de la résolution des IRI relatifs.

Depuis la version 1.2.1.0 du moteur, Neptune utilise `http://aws.amazon.com/neptune/default/` comme IRI de base si aucun IRI de base explicite ne fait partie de la demande.

Dans la demande suivante, l'IRI de base fait partie de la demande :

```
BASE <http://example.org/default/>
INSERT DATA { <node1> <id> "n1" }

BASE <http://example.org/default/>
SELECT * { <node1> ?p ?o }
```

Le résultat serait le suivant :

```
?p                                                  ?o
http://example.org/default/id                       n1
```

En revanche, dans cette demande, aucun IRI de base n'est inclus :

```
INSERT DATA { <node1> <id> "n1" }

SELECT * { <node1> ?p ?o }
```

Dans ce cas, le résultat serait :

```
?p                                                  ?o
http://aws.amazon.com/neptune/default/id            n1
```

Valeurs xsd:dateTime dans Neptune

Pour des raisons de performance, Neptune stocke toujours les date/time valeurs en temps universel coordonné (UTC). Cela rend les comparaisons directes très efficaces.

Cela signifie également que si vous entrez une valeur `dateTime` spécifiant un fuseau horaire particulier, Neptune convertit cette valeur en UTC et rejette ces informations de fuseau horaire.

Ensuite, lorsque vous récupérez la valeur `dateTime` ultérieurement, elle est exprimée en UTC, et non en heure du fuseau horaire d'origine, et vous ne pouvez plus savoir quel était ce fuseau horaire d'origine.

Traitement des valeurs spéciales à virgule flottante par Neptune

Neptune traite les valeurs spéciales à virgule flottante dans SPARQL comme suit :

Gestion des valeurs NaN SPARQL dans Neptune

Dans Neptune, SPARQL peut accepter une valeur `NaN` dans une requête. Aucune distinction n'est faite entre les valeurs `NaN` de signalisation et silencieuses. Neptune considère toutes les valeurs `NaN` comme étant silencieuses.

D'un point de vue sémantique, aucune comparaison d'une valeur `NaN` n'est possible, car aucune valeur n'est supérieure, inférieure ou égale à une valeur `NaN`. Cela signifie qu'une valeur `NaN` d'un côté d'une comparaison ne correspond jamais à quoi que ce soit de l'autre côté.

Cependant, la spécification XSD traite deux valeurs `NaN` `xsd:double` ou `xsd:float` comme égales. Neptune respecte ce principe pour le filtre `IN`, pour l'opérateur égal dans les expressions de filtre, et pour la sémantique de correspondance exacte (ayant un élément `NaN` dans la position de l'objet d'un modèle triple).

Traitement des valeurs infinies SPARQL dans Neptune

Dans Neptune, SPARQL peut accepter une valeur `INF` ou `-INF` dans une requête. Une valeur `INF` est comparée comme étant supérieure à toute autre valeur numérique, et une valeur `-INF` est comparée comme étant inférieure à toute autre valeur numérique.

Deux valeurs INF avec des signes correspondants sont égales l'une à l'autre quel que soit leur type (par exemple, un nombre à virgule flottante `-INF` est égal à un double `-INF`).

Bien sûr, aucune comparaison avec une valeur `NaN` n'est possible, car aucune valeur n'est supérieure, inférieure ou égale à une valeur `NaN`.

Traitement du zéro négatif SPARQL dans Neptune

Neptune normalise une valeur de zéro négatif en un zéro non signé. Des valeurs nulles négatives peuvent être utilisées dans une requête, mais elles ne sont pas enregistrées en tant que telles dans la base de données et elles sont comparées comme étant égales à des zéros non signés.

Limite des valeurs de Neptune Arbitrary-Length

Neptune limite à 64 bits la taille de stockage des valeurs entières, à virgule flottante et décimales XSD dans SPARQL. L'utilisation de valeurs plus grandes génère une erreur `InvalidNumericDataException`.

Neptune étend la comparaison Equals dans SPARQL

La norme SPARQL définit une logique ternaire pour les expressions de valeur, dans lesquelles une expression de valeur peut être évaluée à `true`, `false` ou `error`. La sémantique par défaut pour l'égalité des termes telle que définie dans la spécification SPARQL 1.1, qui s'applique aux comparaisons = et `!=` dans des conditions `FILTER` génère une `error` lors de la comparaison de types de données qui ne sont pas comparables explicitement dans la table des opérateurs dans la spécification.

Ce comportement peut entraîner des résultats non intuitifs, comme dans l'exemple suivant.

Données :

```
<http://example.com/Server/1> <http://example.com/ip> "127.0.0.1"^^<http://example.com/
datatype/IPAddress>
```

Requête 1 :

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o = "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

Requête 2 :

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o != "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

Avec la sémantique SPARQL par défaut utilisée par Neptune avant la version 1.0.2.1, les deux requêtes renvoyaient le résultat vide. En effet, `?o = "127.0.0.2"^^<http://example.com/IPAddress>`, lorsqu'il est évalué pour `?o := "127.0.0.1"^^<http://example.com/IPAddress>`, génère le résultat `error` plutôt que `false`, car aucune règle de comparaison explicite

n'est spécifiée pour le type de données personnalisé <http://example.com/IPAddress>. Par conséquent, la version négative dans la deuxième requête produit également une error. Dans les deux requêtes, error provoque le filtrage de la solution candidate.

À partir de la version 1.0.2.1, Neptune a étendu l'opérateur d'inégalité SPARQL conformément à la spécification. Consultez la section SPARQL 1.1 sur l'extensibilité de l'opérateur, qui permet aux moteurs de définir des règles supplémentaires sur la façon d'effectuer une comparaison entre des types de données intégrés définis par l'utilisateur et non comparables.

En utilisant cette option, Neptune traite désormais une comparaison de deux types de données qui n'est pas explicitement définie dans la table de mappage des opérateurs comme prenant la valeur true si les valeurs littérales et les types de données sont égaux d'un point de vue syntaxique, ou comme prenant la valeur false dans le cas contraire. Un error n'est en aucun cas produit.

En utilisant ces nouvelles sémantiques, la deuxième requête renverrait "127.0.0.1"^^<http://example.com/IPAddress> au lieu d'un résultat vide.

Gestion des Out-of-Range littéraux dans Neptune SPARQL

La sémantique XSD définit chaque type numérique avec son espace de valeurs, à l'exception de integer et decimal. Ces définitions limitent chaque type à une plage de valeurs. Par exemple, la plage d'une plage xsd:byte est comprise entre -128 et +127, inclusivement. Toute valeur en dehors de cette plage est considérée comme non valide.

Si vous essayez d'affecter une valeur littérale en dehors de l'espace de valeurs d'un type (par exemple, si vous essayez de définir xsd:byte sur une valeur littérale de 999), Neptune accepte la valeur hors plage telle quelle, sans l'arrondir ni la tronquer. Mais il ne le persiste pas en tant que valeur numérique car le type donné ne peut pas le représenter.

Autrement dit, Neptune accepte "999"^^xsd:byte même s'il s'agit d'une valeur qui se trouve en dehors de la plage de valeurs xsd:byte définie. Cependant, une fois la valeur conservée dans la base de données, elle ne peut être utilisée que dans la sémantique de correspondance exacte, dans une position d'objet d'un motif triple. Aucun filtre de plage ne peut être exécuté dessus car les littéraux hors plage ne sont pas traités comme des valeurs numériques.

La spécification SPARQL 1.1 définit les opérateurs de plage sous la forme numeric -operator-numeric, string -operator-string, literal-operator-literal, etc. Neptune ne peut pas exécuter un opérateur de comparaison de plages tel que invalid-literal-operator-numeric-value.

Conformité aux spécifications OpenCypher dans Amazon Neptune

La version Amazon Neptune d'openCypher prend généralement en charge les clauses, les opérateurs, les expressions, les fonctions et la syntaxe définis dans la spécification openCypher actuelle, à savoir la version 9 de Cypher Query Language Reference. Les limites et les différences relatives à la prise en charge d'openCypher par Neptune sont décrites ci-dessous.

Amazon Neptune prend également en charge plusieurs fonctionnalités qui dépassent le cadre de la spécification OpenCypher. Pour plus d'informations, consultez Extensions OpenCypher dans Amazon Neptune.

> **Note**
>
> L'implémentation Neo4j actuelle de Cypher contient des fonctionnalités qui ne se trouvent pas dans la spécification openCypher mentionnée ci-dessus. Si vous migrez le code Cypher actuel vers Neptune, consultez Compatibilité de Neptune avec Neo4j et Réécriture des requêtes Cypher pour les exécuter dans openCypher sur Neptune pour plus d'informations.

Prise en charge des clauses openCypher dans Neptune

Neptune prend en charge les clauses suivantes, sauf indication contraire :

- `MATCH` : prise en charge, mais pas *`shortestPath()`* et *`allShortestPaths()`*
- `OPTIONAL MATCH`
- *`MANDATORY MATCH`* : n'est actuellement pas pris en charge dans Neptune. Neptune prend toutefois en charge les valeurs d'ID personnalisées dans les requêtes `MATCH`.
- `RETURN` : pris en charge, sauf lorsqu'il est utilisé avec des valeurs non statiques pour `SKIP` ou `LIMIT`. Par exemple, voici ce qui ne fonctionne pas pour l'instant :

```
MATCH (n)
RETURN n LIMIT toInteger(rand())    // Does NOT work!
```

- `WITH` : pris en charge, sauf lorsqu'il est utilisé avec des valeurs non statiques pour `SKIP` ou `LIMIT`. Par exemple, voici ce qui ne fonctionne pas pour l'instant :

```
MATCH (n)
WITH n SKIP toInteger(rand())
```

```
WITH count() AS count
RETURN count > 0 AS nonEmpty    // Does NOT work!
```

- `UNWIND`
- `WHERE`
- `ORDER BY`
- `SKIP`
- `LIMIT`
- `CREATE` : Neptune vous permet de créer des valeurs d'ID personnalisées dans les requêtes `CREATE`.
- `DELETE`
- `SET`
- `REMOVE`
- `MERGE` : Neptune prend en charge les valeurs d'ID personnalisées dans les requêtes `MERGE`.
- *`CALL[YIELD...]`* : n'est actuellement pas pris en charge dans Neptune.
- `UNION, UNION ALL` : les requêtes en lecture seule sont prises en charge, mais les requêtes de mutation ne le sont pas actuellement.
- `USING`— `USING` est pris en charge depuis la version 1.3.2.0 du moteur. Pour plus d'informations, consultez la section Conseils relatifs aux requêtes.

Prise en charge des opérateurs openCypher dans Neptune

Neptune prend en charge les opérateurs suivants, sauf indication contraire :

Opérateurs généraux

- `DISTINCT`
- L'opérateur **`.`** permettant d'accéder aux propriétés d'un mappage de littéraux imbriqué.

Operateurs mathématiques

- Opérateur d'addition **`+`**.
- L'opérateur de soustraction **`-`**.
- Opérateur de multiplication **`*`**.

- L'opérateur de division **/**.
- L'opérateur de division modulo **%**.
- L'opérateur d'**^**exponentiation. *is NOT supported*

Opérateurs de comparaison

- Opérateur d'addition **=**.
- L'opérateur d'égalité **<>**.
- L'opérateur **<** (inférieur à) est pris en charge sauf lorsque l'un des arguments est un chemin, une liste ou un mappage.
- L'opérateur **>** (supérieur à) est pris en charge sauf lorsque l'un des arguments est un chemin, une liste ou un mappage.
- L'opérateur **<=** (inférieur ou égal à) est pris en charge sauf lorsque l'un des arguments est un chemin, une liste ou un mappage.
- L'opérateur **>=** (inférieur ou supérieur à) est pris en charge sauf lorsque l'un des arguments est un chemin, une liste ou un mappage.
- `IS NULL`
- `IS NOT NULL`
- `STARTS WITH` est pris en charge si les données recherchées correspondant à une chaîne.
- `ENDS WITH` est pris en charge si les données recherchées correspondant à une chaîne.
- `CONTAINS` est pris en charge si les données recherchées correspondant à une chaîne.

Opérateurs booléens

- `AND`
- `OR`
- `XOR`
- `NOT`

Opérateurs de chaîne

- Opérateur de concaténation **+**.

Opérateurs de liste

- Opérateur de concaténation **+**.
- `IN` (vérifie la présence d'un élément dans la liste)

Prise en charge des expressions openCypher dans Neptune

Neptune prend en charge les expressions suivantes, sauf indication contraire :

- `CASE`
- L'expression **`[]`** n'est actuellement pas prise en charge dans Neptune pour accéder aux clés de propriété calculées dynamiquement au sein d'un nœud, d'une relation ou d'un mappage. Par exemple, les éléments suivants ne fonctionnent pas :

```
MATCH (n)
WITH [5, n, {key: 'value'}] AS list
RETURN list[1].name
```

Prise en charge des fonctions openCypher dans Neptune

Neptune prend en charge les fonctions suivantes, sauf indication contraire :

Fonctions de prédicat

- `exists()`

Fonctions scalaires

- `coalesce()`
- `endNode()`
- `epochmillis()`
- `head()`
- `id()`
- `last()`
- `length()`
- `randomUUID()`

- `properties()`
- `removeKeyFromMap`
- `size()` : cette méthode surchargée ne fonctionne actuellement que pour les expressions de modèles, les listes et les chaînes.
- `startNode()`
- `timestamp()`
- `toBoolean()`
- `toFloat()`
- `toInteger()`
- `type()`

Fonctions d'agrégation

- `avg()`
- `collect()`
- `count()`
- `max()`
- `min()`
- `percentileDisc()`
- `stDev()`
- `percentileCont()`
- `stDevP()`
- `sum()`

Répertorier des fonctions

- `join()` (concatène les chaînes d'une liste en une seule chaîne)
- `keys()`
- `labels()`
- `nodes()`
- `range()`

- `relationships()`
- `reverse()`
- `tail()`

Fonctions mathématiques (numériques)

- `abs()`
- `ceil()`
- `floor()`
- `rand()`
- `round()`
- `sign()`

Fonctions mathématiques (logarithmiques)

- `e()`
- `exp()`
- `log()`
- `log10()`
- `sqrt()`

Fonctions mathématiques (trigonométriques)

- `acos()`
- `asin()`
- `atan()`
- `atan2()`
- `cos()`
- `cot()`
- `degrees()`
- `pi()`
- `radians()`

- `sin()`
- `tan()`

Fonctions de chaîne

- `join()` (concatène les chaînes d'une liste en une seule chaîne)
- `left()`
- `lTrim()`
- `replace()`
- `reverse()`
- `right()`
- `rTrim()`
- `split()`
- `substring()`
- `toLower()`
- `toString()`
- `toUpper()`
- `trim()`

User-defined fonctions

*`User-defined functions`*ne sont actuellement pas pris en charge dans Neptune.

Neptune-specific Détails de l'implémentation d'OpenCypher

Les sections suivantes décrivent en quoi l'implémentation Neptune d'openCypher peut différer ou aller au-delà de la spécification openCypher.

Variable-length évaluations du chemin (VLP) dans Neptune

Les évaluations de chemins de longueur variable (VLP) découvrent les chemins entre les nœuds du graphe. La longueur du chemin peut être illimitée dans une requête. Pour éviter les cycles, la spécification openCypher indique que chaque arête doit être traversée au maximum une fois par solution.

Pour les VLP, l'implémentation Neptune s'écarte de la spécification openCypher en ce sens qu'elle ne prend en charge que les valeurs de constantes pour les filtres d'égalité des propriétés. Prenons la requête suivante :

```
MATCH (x)-[:route*1..2 {dist:33, code:x.name}]->(y) return x,y
```

La valeur du filtre d'égalité des propriétés `x.name` n'étant pas une constante, cette requête génère une exception `UnsupportedOperationException` avec le message suivant : `Property predicate over variable-length relationships with non-constant expression is not supported in this release.`

Support temporel dans l'implémentation de Neptune OpenCypher (base de données Neptune 1.3.1.0 et versions antérieures)

Neptune fournit actuellement une prise en charge limitée de la fonction temporelle dans openCypher. Il prend en charge le type de données `DateTime` pour les types temporels.

La fonction `datetime()` peut être utilisée pour obtenir la date et l'heure UTC actuelles comme suit :

```
RETURN  datetime() as res
```

Les valeurs de date et d'heure peuvent être analysées à partir de chaînes dans un format ''dateTheure'' où date et heure sont toutes deux exprimées sous l'une des formes prises en charge ci-dessous :

Formats de date pris en charge

- `yyyy-MM-dd`
- `yyyyMMdd`
- `yyyy-MM`
- `yyyy-DDD`
- `yyyyDDD`
- `yyyy`

Formats pris en charge

- `HH:mm:ssZ`

- `HHmmssZ`
- `HH:mm:ssZ`
- `HH:mmZ`
- `HHmmZ`
- `HHZ`
- `HHmmss`
- `HH:mm:ss`
- `HH:mm`
- `HHmm`
- `HH`

Par exemple :

```
RETURN datetime('2022-01-01T00:01')      // or another example:
RETURN datetime('2022T0001')
```

Notez que toutes les date/time valeurs de Neptune OpenCypher sont stockées et récupérées sous forme de valeurs UTC.

Neptune openCypher utilise une horloge `statement`, ce qui signifie que le même instant dans le temps est utilisé pendant toute la durée d'une requête. Une requête différente au sein de la même transaction peut utiliser un instant différent.

Neptune ne prend pas en charge l'utilisation d'une fonction dans un appel à `datetime()`. Voici, par exemple, ce qui ne fonctionne pas :

```
CREATE (:n {date:datetime(tostring(2021))})  // ---> NOT ALLOWED!
```

Neptune prend en charge la fonction `epochmillis()` qui convertit une valeur `datetime` en `epochmillis`. Par exemple :

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

Neptune ne prend actuellement pas en charge les autres fonctions et opérations au niveau des objets `DateTime`, telles que l'addition et la soustraction.

Support temporel dans l'implémentation de Neptune OpenCypher (Neptune Analytics et Neptune Database 1.3.2.0 et versions ultérieures)

La fonctionnalité date/heure suivante OpenCypher s'applique à Neptune Analytics. Vous pouvez également utiliser le paramètre labmode `DatetimeMillisecond=enabled` pour activer la fonctionnalité datetime suivante sur les versions 1.3.2.0 et supérieures du moteur Neptune. Pour plus de détails sur l'utilisation de cette fonctionnalité en mode laboratoire, consultezSupport date/heure étendu.

- Support en millisecondes. Le littéral Datetime sera toujours renvoyé en millisecondes, même si le nombre de millisecondes est égal à 0. (Le comportement précédent consistait à tronquer des millisecondes.)

```
CREATE (:event {time: datetime('2024-04-01T23:59:59Z')})

# Returning the date returns with 000 suffixed representing milliseconds
MATCH(n:event)
RETURN n.time as datetime

{
  "results" : [ {
    "n" : {
      "~id" : "0fe88f7f-a9d9-470a-bbf2-fd6dd5bf1a7d",
      "~entityType" : "node",
      "~labels" : [ "event" ],
      "~properties" : {
        "time" : "2024-04-01T23:59:59.000Z"
      }
    }
  } ]
}
```

- Support pour appeler la fonction datetime () sur des propriétés stockées ou des résultats intermédiaires. Par exemple, les requêtes suivantes n'étaient pas possibles avant cette fonctionnalité.

  Datetime () sur les propriétés :

```
// Create node with property 'time' stored as string
CREATE (:event {time: '2024-04-01T23:59:59Z'})

// Match and return this property as datetime
```

```
MATCH(n:event)
RETURN datetime(n.time) as datetime
```

Datetime () sur les résultats intermédiaires :

```
// Parse datetime from parameter
UNWIND $list as myDate
RETURN datetime(myDate) as d
```

- Il est désormais également possible de sauvegarder les propriétés date/heure créées dans les cas mentionnés ci-dessus.

  Sauvegarde de la date et de l'heure de la propriété string d'une propriété vers une autre :

```
// Create node with property 'time' stored as string
CREATE (:event {time: '2024-04-01T23:59:59Z', name: 'crash'})

// Match and update the same property to datetime type
MATCH(n:event {name: 'crash'})
SET n.time = datetime(n.time)

// Match and update another node's property
MATCH(e:event {name: 'crash'})
MATCH(n:server {name: e.servername})
SET n.time = datetime(e.time)
```

  Créez des nœuds par lots à partir d'un paramètre doté d'une propriété datetime :

```
// Batch create from parameter
UNWIND $list as events
CREATE (n:crash) {time: datetime(events.time)}
// Parameter value
{
  "x":[
    {"time":"2024-01-01T23:59:29", "name":"crash1"},
    {"time":"2023-01-01T00:00:00Z", "name":"crash2"}
  ]
}
```

- Support d'un plus grand sous-ensemble de formats date/heure ISO8601. Reportez-vous à ci-dessous.

Formats pris en charge

Le format d'une valeur date/heure est [Date] T [Heure] [Fuseau horaire], où T est le séparateur. Si aucun fuseau horaire explicite n'est fourni, UTC (Z) est supposé être le fuseau horaire par défaut.

Fuseau horaire

Les formats de fuseau horaire pris en charge sont les suivants :

- +/-HH:mm
- +/-HHmm
- +/-HH

La présence d'un fuseau horaire dans une chaîne de date/heure est facultative. Si le décalage horaire est de 0, Z peut être utilisé à la place du suffixe de fuseau horaire ci-dessus pour indiquer l'heure UTC. La plage prise en charge d'un fuseau horaire est comprise entre -14h00 et +14h00.

Date

Si aucun fuseau horaire n'est présent ou si le fuseau horaire est UTC (Z), les formats de date pris en charge sont les suivants :

> **Note**
>
> DDD fait référence à une date ordinale, qui représente un jour de l'année compris entre 001 et 365 (366 les années bissextiles). Par exemple, 2024-002 représente le 2 janvier 2024.

- `yyyy-MM-dd`
- `yyyyMMdd`
- `yyyy-MM`
- `yyyyMM`
- `yyyy-DDD`
- `yyyyDDD`
- `yyyy`

Si un fuseau horaire autre que Z est choisi, les formats de date pris en charge sont limités aux suivants :

- `yyyy-MM-dd`
- `yyyy-DDD`
- `yyyyDDD`

La plage de dates prise en charge est comprise entre 1400-01-01 et 9999-12-31.

Heure

Si aucun fuseau horaire n'est présent ou si le fuseau horaire est UTC (Z), les formats horaires pris en charge sont les suivants :

- `HH:mm:ss.SSS`
- `HH:mm:ss`
- `HHmmss.SSS`
- `HHmmss`
- `HH:mm`
- `HHmm`
- `HH`

Si un fuseau horaire autre que Z est choisi, les formats horaires pris en charge sont limités aux suivants :

- `HH:mm:ss`
- `HH:mm:ss.SSS`

Différences de sémantique du langage Neptune openCypher

Neptune représente les ID de nœuds et de relations sous forme de chaînes plutôt que d'entiers. L'ID équivaut à celui fourni par le biais du chargeur de données. S'il existe un espace de noms pour la colonne, il s'agit de l'espace de noms plus l'ID. Par conséquent, la fonction `id` renvoie une chaîne au lieu d'un entier.

Le type de données `INTEGER` est limité à 64 bits. Lors de la conversion de valeurs à virgule flottante ou de valeurs de chaîne plus grandes en entier à l'aide de la fonction `TOINTEGER`, les valeurs négatives sont tronquées en `LLONG_MIN`, et les valeurs positives sont tronquées en `LLONG_MAX`.

Par exemple :

```
RETURN TOINTEGER(2^100)
>  9223372036854775807

RETURN TOINTEGER(-1 * 2^100)
>  -9223372036854775808
```

Multi-valued propriétés

Bien qu'OpenCypher CREATE ne crée pas de propriétés à valeurs multiples, celles-ci peuvent exister dans les données créées à l'aide de Gremlin (base de données Neptune) ou lors du chargement de données (base de données Neptune et Neptune Analytics). Si Neptune openCypher trouve une propriété à valeurs multiples, l'une de ces valeurs est choisie arbitrairement, ce qui crée un résultat non déterministe.

Gestion des valeurs NaN

La façon dont Neptune gère la comparaison des valeurs des `NaN` propriétés n'est pas définie. Le fait de s'appuyer sur de telles comparaisons peut donner des résultats inattendus ou non déterministes.

Modèle de données de graphe de Neptune

L'unité de base des données d'un graphe Amazon Neptune est un élément à quatre positions (quadruplet), qui est comparable à un quadruplet RDF (Resource Description Framework). Voici les quatre positions d'un quadruplet Neptune :

- `subject  (S)`
- `predicate  (P)`
- `object  (O)`
- `graph  (G)`

Chaque quadruplet est une instruction qui effectue une assertion sur une ou plusieurs ressources. Une instruction peut déclarer l'existence d'une relation entre deux ressources, ou elle peut attacher une propriété (paire clé/valeur) à une ressource. Vous pouvez généralement considérer la valeur de prédicat de quadruplet comme le verbe de l'instruction. Elle décrit le type de relation ou la propriété qui est définie. L'objet est la cible de la relation ou la valeur de la propriété. Voici quelques exemples :

- Une relation entre deux sommets peut être représentée en stockant l'identifiant du sommet source dans la position `S`, l'identifiant du sommet cible dans la position `O` et l'étiquette d'arc dans la position `P`.
- Une propriété peut être représentée en stockant l'identifiant d'élément dans la position `S`, la clé de propriété dans la position `P` et la valeur de propriété dans la position `O`.

La position du graphe `G` est utilisée différemment dans les différentes piles. Pour les données RDF dans Neptune, la position `G` contient un identifiant de graphe nommé. Pour les graphes de propriétés dans Gremlin, elle sert à stocker la valeur d'ID d'arête dans le cas d'une arête. Dans tous les autres cas, la valeur par défaut est fixe.

Un graphe est constitué d'un ensemble de déclarations de quadruplet avec des identifiants de ressources partagées.

Dictionnaire des valeurs destinées à l'utilisateur

Neptune ne stocke pas la plupart des valeurs destinées aux utilisateurs directement dans les différents index qu'il gère. Il les stocke plutôt séparément dans un dictionnaire et les remplace dans les index par des identifiants à 8 octets.

- Toutes les valeurs destinées à l'utilisateur qui figureraient dans les index `S`, `P` ou `G` sont stockées dans le dictionnaire de cette manière.
- Dans l'index 0, les valeurs numériques sont stockées directement (de manière intégrée). Cela inclut les valeurs `date` et `datetime` (représentées en millisecondes à partir de l'époque Unix).
- Toutes les autres valeurs destinées à l'utilisateur qui figureraient dans l'index 0 sont stockées dans le dictionnaire et représentées dans l'index par des ID.

Le dictionnaire contient un mappage direct des valeurs destinées à l'utilisateur avec des ID de 8 octets dans un index `value_to_id`.

Il stocke le mappage inverse des ID de 8 octets avec les valeurs de l'un des deux index, en fonction de la taille de ces valeurs :

- Un index `id_to_value` mappe les ID avec les valeurs destinées à l'utilisateur inférieures à 767 octets après l'encodage interne.
- Un index `id_to_blob` mappe les ID avec à les valeurs supérieures destinées à l'utilisateur.

Comment les instructions sont indexées dans Neptune

Lorsque vous interrogez un graphe de quadruplets, vous pouvez ou non spécifier une contrainte de valeur pour chaque position de quadruplet. La requête renvoie tous les quadruplets qui correspondent aux contraintes de valeur que vous avez spécifiées.

Neptune utilise des index pour résoudre les modèles de requêtes graphiques. Ces index concernent les quatre composants principaux d'une arête de graphe : sujet (sommet source dans LPG) ; prédicat (RDF) ou étiquette de propriété ou de bord (LPG) ; objet (sommet cible ou valeur de propriété dans LPG) ; et graphe (RDF) ou identifiant de bord (LPG). Il existe 16 (2^4) modèles d'accès possibles pour ces quatre positions de composants. Vous pouvez interroger efficacement les 16 modèles sans avoir à scanner ni à filtrer à l'aide de six index. Chaque index d'instruction de quadruplet utilise une clé composée des quatre valeurs de position concaténées dans un ordre différent. Voici une combinaison possible de quatre index d'instructions qui couvriraient les 16 chemins d'accès :

```
     Access Pattern                                    Index key order
-------------------------------------------------    ---------------
 1.  ????  (No constraints; returns every quad)           SPOG
 2.  SPOG  (Every position is constrained)                SPOG
```

```
 3.  SPO?  (S, P, and O are constrained; G is not)          SPOG
 4.  SP??  (S and P are constrained; O and G are not)       SPOG
 5.  S???  (S is constrained; P, O, and G are not)          SPOG
 6.  S??G  (S and G are constrained; P and O are not)       SPOG

 7.  ?POG  (P, O, and G are constrained; S is not)          POGS
 8.  ?PO?  (P and O are constrained; S and G are not)       POGS
 9.  ?P??  (P is constrained; S, O, and G are not)          POGS

10.  ?P?G  (P and G are constrained; S and O are not)       GPSO
11.  SP?G  (S, P, and G are constrained; O is not)          GPSO
12.  ???G  (G is constrained; S, P, and O are not)          GPSO

13.  S?OG  (S, O, and G are constrained; P is not)          OGSP
14.  ??OG  (O and G are constrained; S and P are not)       OGSP
15.  ??O?  (O is constrained; S, P, and G are not)          OGSP

16.  S?O?  (S and O are constrained; P and G are not)       OSGP
```

Neptune crée et gère uniquement trois de ces six index par défaut :

- `SPOG` – Utilise une clé composée de `Subject + Predicate + Object + Graph`.
- `POGS` – Utilise une clé composée de `Predicate + Object + Graph + Subject`.
- `GPSO` – Utilise une clé composée de `Graph + Predicate + Subject + Object`.

Ces trois index gèrent la plupart des modèles d'accès les plus courants. Le fait de conserver seulement trois index d'instruction complets au lieu de six a pour effet de limiter grandement les ressources dont vous avez besoin pour prendre en charge l'accès rapide sans analyse et filtrage. Par exemple, l'index `SPOG` permet une recherche efficace chaque fois qu'un préfixe des positions, comme le sommet ou l'identifiant de sommet et de propriété, est lié. L'index `POGS` permet un accès efficace lorsque seule l'étiquette d'arête ou de propriété stockée à la position `P` est liée.

L'API de bas niveau permettant de trouver des instructions utilise un modèle d'instruction dans lequel certaines positions sont connues et les autres doivent être découvertes à l'aide d'une recherche d'index. En constituant les positions connues dans un préfixe de clé en fonction de l'ordre de la clé d'index pour l'un des index d'instruction, Neptune exécute une analyse de plage pour récupérer toutes les instructions correspondant aux positions connues.

Cependant, un des index d'instruction que Neptune ne crée pas par défaut est un index `OSGP` de traversée inverse, qui peut collecter des prédicats dans les objets et les sujets. Au lieu de cela,

Neptune effectue par défaut le suivi des différents prédicats dans un index distinct qu'il utilise pour effectuer une analyse d'union de `{all P x POGS}`. Lorsque vous utilisez Gremlin, un prédicat correspond à une propriété ou à une étiquette d'arête.

Si le nombre de prédicats distincts dans un graphe devient important, la stratégie d'accès Neptune par défaut peut devenir inefficace. Dans Gremlin, par exemple, une étape `in()` dans laquelle aucune étiquette d'arc n'est donnée, ou toute étape qui utilise `in()` en interne, comme `both()` ou `drop()`, peut devenir plutôt inefficace.

Activation de la création d'index OSGP en mode Lab

Si votre modèle de données crée un grand nombre de prédicats distincts, vous pouvez être confronté à des performances réduites et des coûts opérationnels plus élevés qui peuvent être considérablement améliorés en utilisant le mode laboratoire pour activer l'index OSGP, en plus des trois index qui sont gérés par défaut par Neptune.

L'activation de l'index OSGP peut présenter quelques inconvénients :

- Le taux d'insertion peut être ralenti de jusqu'à 23 %.
- Le stockage augmente jusqu'à 20 %.
- Les requêtes de lecture qui touchent tous les index de manière égale (ce qui est plutôt rare) peuvent avoir des latences accrues.

En général, cependant, il vaut la peine d'activer l'index OSGP pour les clusters de base de données comportant un grand nombre de prédicats distincts. Object-based les recherches deviennent très efficaces (par exemple, trouver toutes les arêtes entrantes d'un sommet, ou tous les sujets connectés à un objet donné), et par conséquent, la suppression de sommets devient également beaucoup plus efficace.

> **Important**
>
> Vous ne pouvez activer l'index OSGP que dans un cluster de bases de données vide, avant d'y charger des données.

Déclarations Gremlin dans le modèle de données Neptune

Les données du graphe de propriétés Gremlin sont exprimées dans le modèle SPOG à l'aide de trois classes de déclarations, à savoir :

- Instructions d'étiquette de sommet
- Instructions d'arc
- Instructions de propriété

Pour en savoir plus sur leur utilisation dans les requêtes Gremlin, consultez Présentation du fonctionnement des requêtes Gremlin dans Neptune.

Le cache de recherche Neptune peut accélérer les requêtes de lecture

Amazon Neptune implémente un cache de recherche qui utilise le NVMe-based SSD de l'`R5d`instance pour améliorer les performances de lecture des requêtes impliquant des recherches fréquentes et répétitives de valeurs de propriétés ou de littéraux RDF. Le cache de recherche stocke temporairement ces valeurs dans le volume SSD NVMe où elles sont rapidement accessibles.

Les requêtes de lecture qui renvoient les propriétés d'un grand nombre de sommets et d'arêtes, ou de nombreux triplets RDF, peuvent avoir une latence élevée si les valeurs des propriétés ou les littéraux doivent être extraits des volumes de stockage du cluster plutôt que de la mémoire. Comme exemple, citons les requêtes de lecture de longue durée qui renvoient un grand nombre de noms complets à partir d'un graphe d'identité ou un grand nombre d'adresses IP à partir d'un graphe de détection de fraude. Plus le nombre de valeurs de propriétés ou de littéraux RDF renvoyés par la requête augmente, plus la mémoire disponible diminue, et l'exécution de la requête peut se dégrader de manière significative.

Cas d'utilisation du cache de recherche Neptune

Le cache de recherche n'est utile que lorsque les requêtes de lecture renvoient les propriétés d'un très grand nombre de sommets et d'arêtes, ou de triplets RDF.

Pour optimiser les performances des requêtes, Amazon Neptune utilise le type d'instance `R5d` afin de créer un cache volumineux pour ces valeurs de propriété ou ces littéraux. Leur extraction depuis le cache est alors beaucoup plus rapide que leur extraction depuis des volumes de stockage en cluster.

En règle générale, il ne vaut la peine d'activer le cache de recherche que si les trois conditions suivantes sont remplies :

- Vous avez observé une latence accrue des requêtes de lecture.
- Vous observez également une baisse de la `BufferCacheHitRatio` CloudWatch métrique lors de l'exécution de requêtes de lecture (voirSurveillance de Neptune à l'aide d'Amazon CloudWatch).
- Vos requêtes de lecture passent beaucoup de temps à matérialiser les valeurs renvoyées avant d'afficher les résultats (voir l' Gremlin-profile exemple ci-dessous pour savoir comment déterminer le nombre de valeurs de propriétés qui sont matérialisées pour une requête).

> **Note**
>
> Cette fonctionnalité n'est utile que dans le scénario spécifique décrit ci-dessus. Par exemple, le cache de recherche ne facilite pas du tout les requêtes d'agrégation. À moins que vous n'exécutiez des requêtes qui bénéficieraient du cache de recherche, il n'y a aucune raison d'utiliser un type d'instance R5d au lieu d'un type d'instance R5 équivalent et moins coûteux.

Si vous utilisez Gremlin, vous pouvez évaluer les coûts de matérialisation d'une requête à l'aide de l'API API Gremlin `profile`. La section « Opérations d'index » indique le nombre de termes matérialisés lors de l'exécution :

```
Index Operations
Query execution:
    # of statement index ops: 3
    # of unique statement index ops: 3
    Duplication ratio: 1.0
    # of terms materialized: 5273
Serialization:
    # of statement index ops: 200
    # of unique statement index ops: 140
    Duplication ratio: 1.43
    # of terms materialized: 32693
```

Le nombre de termes non numériques matérialisés est directement proportionnel au nombre de recherches de termes que Neptune doit effectuer.

Utilisation du cache de recherche

Le cache de recherche n'est disponible que sur un type d'instance R5d, où il est automatiquement activé par défaut. Les R5d instances Neptune ont les mêmes spécifications que les R5 instances, plus jusqu'à 1,8 To de NVMe-based stockage SSD local. Les caches de recherche sont spécifiques aux instances, et les charges de travail qui en bénéficient peuvent être dirigées spécifiquement vers les instances R5d d'un cluster Neptune, tandis que les autres charges de travail peuvent être dirigées vers des types d'instances R5 ou autres.

Pour utiliser le cache de recherche sur une instance Neptune, il suffit de mettre à niveau cette dernière vers le type d'instance R5d. Lorsque vous le faites, Neptune définit automatiquement le paramètre du neptune_lookup_cache cluster de base de données sur 1 (activé) et crée le cache de

recherche sur cette instance particulière. Vous pouvez ensuite utiliser l'API Statut d'une instancepour confirmer que le cache a été activé.

De même, pour désactiver le cache de recherche sur une instance donnée, mettez à l'échelle l'instance en la faisant passer d'un type d'instance R5d à un type d'instance R5 équivalent.

Lorsqu'une instance R5d est lancée, le cache de recherche est activé et en mode démarrage à froid, ce qui signifie qu'il est vide. Neptune vérifie d'abord la présence dans le cache de recherche de valeurs de propriétés ou de littéraux RDF lors du traitement des requêtes, puis les ajoute s'ils ne s'y trouvent pas encore. Cela prépare progressivement le cache.

Lorsque vous dirigez les requêtes de lecture qui nécessitent une valeur de propriété ou RDF-literal des recherches vers une instance de lecteur R5d, les performances de lecture se dégradent légèrement pendant le préchauffage de son cache. Cependant, lorsque le cache est réchauffé, les performances de lecture s'accélèrent de manière significative et vous pouvez également constater une baisse des I/O coûts liés aux recherches effectuées dans le cache plutôt qu'au stockage en cluster. L'utilisation de la mémoire s'améliore également.

Si votre instance d'enregistreur est une instance R5d, elle prépare automatiquement son cache de recherche à chaque opération d'écriture. Cette approche augmente légèrement la latence pour les requêtes d'écriture, mais prépare le cache de recherche de manière plus efficace. Ensuite, si vous dirigez les requêtes de lecture qui nécessitent une valeur de propriété ou RDF-literal des recherches vers l'instance du rédacteur, vous commencez à obtenir des performances de lecture améliorées immédiatement, car les valeurs y ont déjà été mises en cache.

En outre, si vous exécutez le chargeur en bloc sur une instance d'enregistreur R5d, vous remarquerez peut-être que ses performances sont légèrement dégradées à cause du cache.

Le cache de recherche étant spécifique à chaque nœud, le remplacement de l'hôte réinitialise le cache à froid.

Vous pouvez désactiver temporairement le cache de recherche sur toutes les instances de votre cluster de base de données en définissant le paramètre de neptune_lookup_cache cluster de base de données sur `0` (désactivé). En général, il est toutefois plus judicieux de désactiver le cache sur des instances spécifiques en les faisant passer des types d'instance R5d à des types d'instance R5 inférieurs.

Sémantique des transactions dans Neptune

Amazon Neptune est conçu pour permettre les charges de travail à traitement transactionnel en ligne (OLTP) hautement simultanées sur les graphes de données. La spécification du langage de requête SPARQL pour RDF du W3C et la documentation du langage de traversée des graphes Apache TinkerPop Gremlin ne définissent pas la sémantique des transactions pour le traitement simultané des requêtes. Comme la prise en charge d'ACID et les garanties de transactions bien définies peuvent être très importantes, nous appliquons une sémantique stricte pour contribuer à éviter les anomalies de données.

Cette section définit ces règles sémantiques et illustre la façon dont elle s'appliquent à certains cas d'utilisation courants dans Neptune.

Rubriques

- Définition des niveaux d'isolement
- Niveaux d'isolement des transactions dans Neptune
- Exemples de sémantique des transactions Neptune

Définition des niveaux d'isolement

Le « I » dans `ACID` signifie isolation. Le degré d'isolement d'une transaction détermine dans quelle mesure les autres transactions simultanées peuvent affecter les données sur lesquelles elle opère.

La SQL:1992norme a créé un vocabulaire pour décrire les niveaux d'isolement. Elle définit trois types d'interactions (que l'on appelle des phénomènes) qui peuvent se produire entre deux transactions simultanées, `Tx1` et `Tx2` :

- `Dirty read` : se produit lorsque `Tx1` modifie un élément, puis que `Tx2` lit cet élément avant que `Tx1` n'ait validé la modification. Ensuite, si `Tx1` ne réussit jamais à valider la modification ou l'annule, `Tx2` a lu une valeur qui n'est jamais parvenue à la base de données.
- `Non-repeatable read` : se produit lorsque `Tx1` lit un élément, puis que `Tx2` modifie ou supprime cet élément et valide la modification, et que `Tx1` essaie de relire l'élément. `Tx1` lit alors une valeur différente de la valeur précédente ou constate que l'élément n'existe plus.
- `Phantom read` : se produit lorsque `Tx1` lit un ensemble d'éléments qui répondent à un critère de recherche, puis que `Tx2` ajoute un nouvel élément qui répond au critère de recherche et que `Tx1` répète la recherche. `Tx1` obtient alors un ensemble d'éléments différent de celui obtenu auparavant.

Chacun de ces trois types d'interaction peut entraîner des incohérences dans les données de résultat d'une base de données.

La SQL:1992 norme a défini quatre niveaux d'isolation qui présentent des garanties différentes en termes de trois types d'interaction et d'incohérences qu'ils peuvent produire. À ces quatre niveaux, une transaction peut avoir la garantie de s'exécuter complètement ou de ne pas s'exécuter du tout :

- `READ UNCOMMITTED` : autorise les trois types d'interaction, c'est-à-dire les lectures corrompues (dirty read), les lectures non reproductibles (non-repeatable read) et les lectures fantôme (t=phantom read).
- `READ COMMITTED` : les lectures corrompues ne sont pas possibles, mais les lectures non reproductibles et fantôme le sont.
- `REPEATABLE READ` : ni les lectures corrompues ni les lectures non reproductibles ne sont possibles, mais les lectures fantôme le sont encore.
- `SERIALIZABLE` : aucun des trois types de phénomène d'interaction ne peut se produire.

Le contrôle de simultanéité multiversion (MVCC) permet un autre type d'isolement, à savoir l'isolement SNAPSHOT . Cela garantit qu'une transaction s'exécute sur un instantané des données telles qu'elles existent lorsque la transaction commence, et qu'aucune autre transaction ne peut modifier cet instantané.

Niveaux d'isolement des transactions dans Neptune

Amazon Neptune implémente différents niveaux d'isolement des transactions pour les requêtes en lecture seule et les requêtes de mutation. Les requêtes SPARQL et Gremlin sont classées comme des requêtes en lecture seule ou de mutation en fonction des critères suivants :

- Dans SPARQL, il existe une distinction claire entre les requêtes de lecture (`SELECT`, `ASK`, `CONSTRUCT` et `DESCRIBE` telles que définies dans la spécification SPARQL 1.1 Query Language) et les requêtes de mutation (`INSERT` et `DELETE` telles que définies dans la spécification SPARQL 1.1 Update).

  Notez que Neptune traite plusieurs requêtes de mutation soumises ensemble (par exemple, dans un message `POST`, en les séparant par des points-virgules) comme une seule transaction. Elles réussiront ou échoueront nécessairement en tant qu'unité atomique, et en cas de défaillance, les changements partiels sont annulés.

- Toutefois, dans Gremlin, Neptune classe une requête comme requête en lecture seule ou requête de mutation selon qu'elle contient ou non des étapes de chemin de requête telles que `addE()`, `addV()`, `property()` ou `drop()` (étapes de manipulation des données). Si la requête contient une étape de ce type, elle est classée et exécutée en tant que requête de mutation.

Il est également possible d'utiliser des sessions permanentes dans Gremlin. Pour de plus amples informations, veuillez consulter Sessions basées sur des scripts Gremlin. Dans ces sessions, toutes les requêtes, y compris les requêtes en lecture seule, sont exécutées de la même manière que les requêtes de mutation sur le point de terminaison de l'enregistreur.

À l'aide des sessions de lecture-écriture Bolt dans openCypher, toutes les requêtes, y compris les requêtes en lecture seule, sont exécutées de la même manière que les requêtes de mutation, sur le point de terminaison de l'enregistreur.

Rubriques

- Read-only isolation des requêtes dans Neptune
- Isolement des requêtes de mutation dans Neptune
- Résolution des conflits à l'aide de Lock-Wait délais
- Verrouillages de plage et faux conflits

Read-only isolation des requêtes dans Neptune

Neptune évalue les requêtes en lecture seule en se conformant à la sémantique d'isolement d'instantané. En d'autres termes, une requête en lecture seule fonctionne logiquement sur un instantané cohérent de la base de données pris au début de l'évaluation de la requête. Neptune peut ainsi garantir qu'aucun des phénomènes suivants ne se produira :

- `Dirty reads`— Read-only les requêtes dans Neptune ne verront jamais de données non validées provenant d'une transaction simultanée.
- `Non-repeatable reads` : une transaction en lecture seule qui lit les mêmes données plusieurs fois revient toujours aux mêmes valeurs.
- `Phantom reads` : une transaction en lecture seule ne lit jamais les données ajoutées après le début de la transaction.

Comme l'isolement des instantanés est obtenu à l'aide du contrôle de simultanéité multiversion (MVCC), les requêtes en lecture seule n'ont pas besoin de verrouiller les données et ne bloquent donc pas les requêtes de mutation.

Les réplicas en lecture acceptent uniquement les requêtes en lecture seule, de sorte que toutes les requêtes portant sur les réplicas en lecture s'exécutent en suivant la sémantique d'isolement `SNAPSHOT`.

La seule considération supplémentaire à prendre en compte lors de l'interrogation d'un réplica en lecture est qu'il peut y avoir un léger retard de réplication entre l'enregistreur et les réplicas en lecture. Cela signifie que la propagation d'une mise à jour effectuée sur l'enregistreur vers le réplica en lecture à partir duquel est effectuée la lecture peut prendre un peu de temps. Le temps de réplication réel dépend de la charge d'écriture par rapport à l'instance principale. L'architecture Neptune prend en charge la réplication à faible latence et le délai de réplication est instrumenté dans une métrique Amazon. CloudWatch

Toutefois, en raison du niveau d'isolement de `SNAPSHOT`, les requêtes de lecture voient toujours un état cohérent de la base de données, même si ce n'est pas le plus récent.

Si vous avez besoin d'être strictement sûr qu'une requête observera le résultat d'une mise à jour précédente, envoyez la requête au point de terminaison de l'enregistreur lui-même plutôt qu'à un réplica en lecture.

Isolement des requêtes de mutation dans Neptune

Les lectures effectuées dans le cadre des requêtes de mutation sont exécutées selon l'isolement de transaction `READ COMMITTED`, ce qui exclut la possibilité de lectures corrompues. Au-delà des garanties habituelles fournies pour l'isolement de transaction `READ COMMITTED`, Neptune offre la garantie qu'aucune lecture `PHANTOM` ou `NON-REPEATABLE` ne pourra se produire.

Ces garanties élevées sont obtenues en verrouillant les enregistrements et les plages d'enregistrements lors de la lecture des données. Cela empêche les transactions simultanées d'effectuer des insertions ou des suppressions dans les plages d'index après leur lecture, ce qui garantit des lectures reproductibles.

> **Note**
>
> Toutefois, une transaction de mutation simultanée `Tx2` peut commencer après le début de la transaction de mutation `Tx1` et peut valider une modification avant que `Tx1` n'ait verrouillé

les données pour les lire. Dans ce cas, `Tx1` voit la modification de `Tx2` comme si `Tx2` avait terminé son opération avant que `Tx1` ne commence. Etant donné que cela s'applique uniquement aux modifications validées, une opération `dirty read` ne peut jamais se produire.

Pour comprendre le mécanisme de verrouillage utilisé par Neptune pour les requêtes de mutation, il est d'abord utile de comprendre les détails du Modèle de données de graphe Neptune et de la Stratégie d'indexation. Neptune gère les données à l'aide de trois index, à savoir `SPOG`, `POGS` et `GPSO`.

Pour obtenir des lectures reproductibles pour le niveau de transaction `READ COMMITTED`, Neptune a des verrouillages de plage dans l'index utilisé. Par exemple, si une requête de mutation lit toutes les propriétés et les arêtes sortantes d'un sommet nommé `person1`, le nœud verrouille toute la plage définie par le préfixe `S=person1` dans l'index `SPOG` avant de lire les données.

Le même mécanisme s'applique lors de l'utilisation des autres index. Par exemple, lorsqu'une transaction de mutation recherche toutes les paires de sommets source-cible pour une étiquette d'arc donnée à l'aide de l'index `POGS`, la plage de l'étiquette d'arc dans la position `P` est verrouillée. Toute transaction simultanée, qu'il s'agisse d'une requête en lecture seule ou d'une requête de mutation, peut malgré tout effectuer des lectures dans la plage verrouillée. Cependant, toute mutation impliquant l'insertion ou la suppression de nouveaux enregistrements dans la plage de préfixes verrouillée nécessite un verrou exclusif et est empêchée.

En d'autres termes, lorsqu'une plage de l'index a été lue par une transaction de mutation, vous bénéficiez de la garantie que cette plage ne sera pas modifiée par des transactions simultanées jusqu'à la fin de la transaction de lecture. Cela garantit qu'aucun événement `non-repeatable reads` ne se produira.

Résolution des conflits à l'aide de Lock-Wait délais

Si une deuxième transaction tente de modifier un enregistrement dans une plage qu'une première transaction a verrouillée, Neptune détecte le conflit immédiatement et bloque la deuxième transaction.

Si aucun blocage de dépendance n'est détecté, Neptune applique automatiquement un mécanisme de délai d'attente de verrouillage, dans lequel la transaction bloquée attend jusqu'à 60 secondes que la transaction contenant le verrouillage se termine et libère le verrouillage.

- Si le délai d'attente de verrouillage expire avant la libération du verrouillage, la transaction bloquée est annulée.
- Si le verrouillage est libéré dans le délai d'attente de verrouillage, la deuxième transaction est débloquée et peut se terminer avec succès sans avoir besoin d'une nouvelle tentative.

Toutefois, si Neptune détecte un blocage de dépendance entre les deux transactions, la résolution automatique du conflit n'est pas possible. Dans ce cas, Neptune annule et restaure immédiatement la deuxième transaction sans initier de délai d'attente de verrouillage. Neptune fait de son mieux pour annuler la transaction ayant le moins d'enregistrements insérés ou supprimés.

Mesure du temps d'attente avant le verrouillage (moteur ≥ 1.4.5.0)

À partir de la version 1.4.5.0 du moteur, vous pouvez observer exactement pendant combien de temps une requête de mutation a été bloquée en utilisant deux compteurs de log des requêtes lentes :

Compteur	Description
`sharedLocksWaitTimeMillis`	Temps passé à attendre pour obtenir des verrous partagés (S), qui autorisent plusieurs lecteurs mais bloquent les rédacteurs.
`exclusiveLocksWaitTimeMillis`	Temps passé à attendre pour obtenir des verrous (X) exclusifs, qui bloquent tous les autres accès.

Ces deux champs apparaissent dans l'`storageCounters`objet uniquement lorsque vous activez le `debug` mode de journalisation par requêtes lentes ()`neptune_enable_slow_query_log=debug`.

> **Tip**
>
> Si elle s'`sharedLocksWaitTimeMillis + exclusiveLocksWaitTimeMillis`approche de celle de la requête`overallRunTimeMs`, celle-ci est bloquée par un problème de verrouillage plutôt que par le processeur, le réseau ou. I/O

Conseils pratiques pour réduire le contentieux :

- Étalez les tâches conflictuelles : exécutez de lourdes mutations par lots pendant les périodes de faible activité des utilisateurs.

- Divisez les grandes mutations en petits morceaux : les petites transactions sont bloquées pendant moins de temps, ce qui réduit le risque de délais d'attente.

Verrouillages de plage et faux conflits

Neptune verrouille les plages à l'aide du verrouillage d'écart. Ce dernier verrouille l'écart entre des enregistrements d'index, ou l'écart avant le premier ou après le dernier enregistrement d'index.

Neptune utilise ce que l'on appelle une table de dictionnaire pour associer des valeurs d'ID numériques à des littéraux de chaîne spécifiques. Voici un exemple d'état d'un tel dictionnaire Neptune :

String	ID
type	1
graphe_par_défaut	2
personne_3	3
personne_1	5
connaît	6
personne_2	7
age	8
arête_1	9
habite_à	10
New York	11
Personne	12
Lieu	13
arête_2	14

Les chaînes ci-dessus appartiennent à un modèle de graphe de propriétés, mais les concepts s'appliquent également à tous les modèles de graphes RDF.

L'état correspondant de l'indice SPOG (Subject-Predicate-Object_Graph) est indiqué ci-dessous à gauche. Sur la droite, les chaînes correspondantes sont affichées pour facilier la compréhension des données d'index.

S (ID)	P (ID)	O (ID)	G (ID)		S (chaîne)	P (chaîne)	O (chaîne)	G (chaîne)
3	1	12	2		personne_3	type	Personne	graphe_par_défaut
5	1	12	2		personne_1	type	Personne	graphe_par_défaut
5	6	3	9		personne_1	connaît	personne_3	arête_1
5	8	40	2		personne_1	age	40	graphe_par_défaut
5	10	11	14		personne_1	habite_à	New York	arête_2
7	1	12	2		personne_2	type	Personne	graphe_par_défaut
11	1	13	2		New York	type	Lieu	graphe_par_défaut

Par exemple, si une requête de mutation lit toutes les propriétés et les arêtes sortantes d'un sommet nommé `person_1`, le nœud verrouille toute la plage définie par le préfixe `S=person_1` dans l'index SPOG avant de lire les données. Le verrouillage de plage verrouillerait l'écart sur tous les enregistrements correspondants et sur le premier enregistrement qui ne correspond pas. Les enregistrements correspondants seraient verrouillés, et les enregistrements restants ne seraient pas verrouillés. Neptune verrouillerait les écarts comme suit :

- `5 1 12 2` (écart 1)

- `5 6 3 9` (écart 2)
- `5 8 40 2` (écart 3)
- `5 10 11 14` (écart 4)
- `7 1 12 2` (écart 5)

Cela verrouille les enregistrements suivants :

- `5 1 12 2`
- `5 6 3 9`
- `5 8 40 2`
- `5 10 11 14`

Dans cet état, les opérations suivantes sont légitimement bloquées :

- Insertion d'une nouvelle propriété ou d'une nouvelle arête pour `S=person_1`. Une nouvelle propriété autre que `type` ou une nouvelle arête devrait être ajoutée dans l'écart 2, l'écart 3, l'écart 4 ou l'écart 5, qui sont tous verrouillés.
- Suppression de l'un des enregistrements existants.

Dans le même temps, quelques opérations simultanées seraient bloquées faussement (générant ainsi de faux conflits) :

- Toutes les insertions de propriétés ou d'arêtes pour `S=person_3` sont bloquées, car elles devraient avoir lieu dans l'écart 1.
- Toute nouvelle insertion de sommet à laquelle est attribué un ID compris entre 3 et 5 serait bloquée, car elle devrait avoir lieu dans l'écart 1.
- Toute nouvelle insertion de sommet à laquelle est attribué un ID compris entre 5 et 7 serait bloquée, car elle devrait avoir lieu dans l'écart 5.

Les verrouillages d'écart ne sont pas suffisamment précis pour verrouiller l'écart correspondant à un prédicat spécifique (par exemple, pour verrouiller l'écart 5 pour le prédicat `S=5`).

Les verrouillages de plage ne sont placés que dans l'index où la lecture a lieu. Dans le cas ci-dessus, les enregistrements sont verrouillés uniquement dans l'index SPOG, pas dans POGS ni GPSO. Les

lectures d'une requête peuvent être effectuées sur tous les index en fonction des modèles d'accès, qui peuvent être répertoriés à l'aide des API `explain` (pour Sparql et pour Gremlin).

> **Note**
>
> Des verrouillages d'écart peuvent également être utilisés pour des mises à jour simultanées sécurisées sur les index sous-jacents, ce qui peut également entraîner de faux conflits. Ces verrouillages d'écart sont placés indépendamment du niveau d'isolement ou des opérations de lecture effectuées par la transaction.

De faux conflits peuvent se produire non seulement lorsque des transactions simultanées entrent en collision en raison de verrouillages d'écart, mais également dans certains cas lorsqu'une transaction fait l'objet d'une nouvelle tentative après un échec quelconque. Si l'annulation déclenchée par l'échec est toujours en cours et que les verrouillages précédemment effectués pour la transaction n'ont pas encore été complètement déverrouillés, la nouvelle tentative donnera lieu à un faux conflit et échouera.

Sous une charge élevée, il est courant que 3 à 4 % des requêtes d'écriture échouent en raison de faux conflits. Pour un client externe, ces faux conflits sont difficiles à prévoir et doivent être gérés à l'aide de nouvelles tentatives.

Exemples de sémantique des transactions Neptune

Les exemples suivants illustrent différents cas d'utilisation de la sémantique des transactions dans Amazon Neptune.

Rubriques

- Exemple 1 : insertion d'une propriété uniquement si elle n'existe pas
- Exemple 2 : assertion selon laquelle une valeur de propriété est globalement unique
- Exemple 3 : modification d'une propriété si une autre propriété a une valeur spécifiée
- Exemple 4 : remplacement d'une propriété existante
- Exemple 5 : éviter les propriétés ou les arêtes sans pendant

Exemple 1 : insertion d'une propriété uniquement si elle n'existe pas

Supposons que vous souhaitiez vous assurer qu'une propriété ne sera définie qu'une seule fois. Par exemple, supposons que plusieurs requêtes tentent simultanément d'attribuer une cote de crédit à une personne. Vous souhaitez n'insérer qu'une seule instance de la propriété et que les autres requêtes échouent, car la propriété a déjà été définie.

```
# GREMLIN:
g.V('person1').hasLabel('Person').coalesce(has('creditScore'), property('creditScore',
 'AAA+'))

# SPARQL:
INSERT { :person1 :creditScore "AAA+" .}
WHERE  { :person1 rdf:type :Person .
         FILTER NOT EXISTS { :person1 :creditScore ?o .} }
```

L'étape Gremlin `property()` insère une propriété avec la clé et la valeur données. L'étape `coalesce()` exécute le premier argument de la première étape, et si elle échoue, elle exécute la deuxième étape :

Avant d'insérer la valeur de la propriété `creditScore` pour un sommet `person1` donné, une transaction doit essayer de lire la valeur `creditScore` potentiellement inexistante pour `person1`. Cette tentative de lecture verrouille la plage SP pour `S=person1` et `P=creditScore` dans l'index SPOG où la valeur `creditScore` se trouve ou sera écrite.

Le fait d'effectuer ce verrouillage de plage empêche toute transaction simultanée d'insérer simultanément une valeur `creditScore`. Lorsqu'il y a plusieurs transactions parallèles, une seule d'entre elles peut mettre à jour la valeur à un moment donné. Cela exclut l'anomalie consistant en la création de plusieurs propriétés `creditScore`.

Exemple 2 : assertion selon laquelle une valeur de propriété est globalement unique

Supposons que vous souhaitiez insérer une personne avec un numéro de sécurité sociale comme clé primaire. Vous souhaitez que votre requête de mutation garantisse que, au niveau global, personne d'autre dans la base de données n'a le même numéro de sécurité sociale :

```
# GREMLIN:
g.V().has('ssn', 123456789).fold()
  .coalesce(__.unfold(),
            __.addV('Person').property('name', 'John Doe').property('ssn', 123456789'))
```

```
# SPARQL:
INSERT { :person1 rdf:type :Person .
         :person1 :name "John Doe" .
         :person1 :ssn 123456789 .}
WHERE  { FILTER NOT EXISTS { ?person :ssn 123456789 } }
```

Cet exemple est similaire au précédent. La principale différence est que le verrouillage de plage est effectué sur l'index P0GS au lieu de l'index SP0G.

La transaction qui exécute la requête doit lire le modèle, `?person :ssn 123456789`, dans lequel les positions 0 et P sont liées. Le verrouillage de plage est effectué sur l'index P0GS pour `P=ssn` et `0=123456789`.

- Si le modèle existe, aucune action n'est effectuée.
- S'il n'existe pas, le verrouillage empêche toute transaction simultanée d'insérer également ce numéro de sécurité sociale.

Exemple 3 : modification d'une propriété si une autre propriété a une valeur spécifiée

Supposons que divers événements d'un jeu déplacent une personne du niveau 1 au niveau 2 et lui attribue une nouvelle propriété `level2Score` définie sur zéro. Vous devez vous assurer que plusieurs instances simultanées de cette transaction ne puissent pas créer plusieurs instances de la propriété de score de niveau 2. Les requêtes dans Gremlin et SPARQL peuvent ressembler à ce qui suit.

```
# GREMLIN:
g.V('person1').hasLabel('Person').has('level', 1)
 .property('level2Score', 0)
 .property(Cardinality.single, 'level', 2)

# SPARQL:
DELETE { :person1 :level 1 .}
INSERT { :person1 :level2Score 0 .
         :person1 :level 2 .}
WHERE  { :person1 rdf:type :Person .
         :person1 :level 1 .}
```

Dans Gremlin, lorsque `Cardinality.single` est spécifié, l'étape `property()` ajoute une nouvelle propriété ou remplace une valeur de propriété existante par la nouvelle valeur spécifiée.

Toute mise à jour d'une valeur de propriété, telle que le passage de la valeur de `level` de 1 à 2, est implémentée sous la forme d'une suppression de l'enregistrement actuel et de l'insertion d'un nouvel enregistrement avec la nouvelle valeur de propriété. Dans ce cas, l'enregistrement avec le numéro de niveau 1 est supprimé et un enregistrement avec le numéro de niveau 2 est réinséré.

Pour que la transaction puisse ajouter `level2Score` et mettre à jour la valeur de `level` de 1 à 2, elle doit d'abord valider le fait que la valeur de `level` est actuellement égale à 1. Pour ce faire, elle effectue un verrouillage de plage sur le préfixe SP0 pour `S=person1`, `P=level` et `0=1` dans l'index SP0G. Ce verrouillage empêche les transactions simultanées de supprimer le triplet de version 1. Par conséquent, aucune mise à jour simultanée conflictuelle ne peut se produire.

Exemple 4 : remplacement d'une propriété existante

Certains événements peuvent mettre à jour la cote de crédit d'une personne en la remplaçant par une nouvelle valeur (ici, BBB). Mais vous voulez vous assurer que les événements simultanés de ce type ne puissent pas créer plusieurs propriétés de cote de crédit pour une personne.

```
# GREMLIN:
g.V('person1').hasLabel('Person')
 .sideEffect(properties('creditScore').drop())
 .property('creditScore', 'BBB')

# SPARQL:
DELETE { :person1 :creditScore ?o .}
INSERT { :person1 :creditScore "BBB" .}
WHERE  { :person1 rdf:type :Person .
         :person1 :creditScore ?o .}
```

Ce cas est similaire à l'exemple 3, sauf qu'au lieu de verrouiller le préfixe SP0, Neptune verrouille le préfixe `P=creditScore` avec SP et `S=person1` uniquement. Cela empêche les transactions simultanées d'insérer ou de supprimer des triplets avec la propriété `creditScore` pour le sujet `person1`.

Exemple 5 : éviter les propriétés ou les arêtes sans pendant

La mise à jour d'une entité ne doit pas laisser un élément sans pendant, c'est-à-dire une propriété ou un arc associé à une entité inexistante. Il s'agit uniquement d'un problème dans SPARQL, car Gremlin a des contraintes intégrées permettant d'empêcher l'existence d'éléments sans pendants.

```
# SPARQL:
```

```
tx1: INSERT { :person1 :age 23 } WHERE { :person1 rdf:type :Person }
tx2: DELETE { :person1 ?p ?o }
```

La requête INSERT doit lire et verrouiller le préfixe SPO avec S=person1, P=rdf:type et O=Person dans l'index SPOG. Le verrouillage empêche la requête DELETE de réussir en parallèle.

Dans la course entre la requête DELETE qui tente de supprimer l'enregistrement :person1 rdf:type :Person et la requête INSERT qui lit l'enregistrement et crée un verrouillage de plage sur son élément SPO dans l'index SPOG, les résultats possibles sont les suivants :

- Si la requête INSERT est validée avant que la requête DELETE ne lise et supprime tous les enregistrements pour :person1, :person1 est entièrement supprimé de la base de données, y compris l'enregistrement nouvellement inséré.
- Si la requête DELETE est validée avant que la requête INSERT n'essaie de lire l'enregistrement :person1 rdf:type :Person, la lecture constate que la modification est validée. Autrement dit, elle ne trouve aucun enregistrement :person1 rdf:type :Person et devient donc un instruction non-opérationnelle.
- Si la requête INSERT est lue avant la requête DELETE, le triplet :person1 rdf:type :Person est verrouillé et la requête DELETE est bloquée jusqu'à ce que la requête INSERT soit validée, comme dans le premier cas précédemment.
- Si la requête DELETE est lue avant la requête INSERT et que la requête INSERT tente de lire et de verrouiller le préfixe SPO de l'enregistrement, un conflit est détecté. Cela est dû au fait que le triplet a été marqué pour suppression et que la requête INSERT a ensuite échoué.

Dans toutes ces différentes séquences d'événements possibles, aucun arc sans pendant n'est créé.

Clusters de bases de données et instances de base de données Amazon Neptune

Un cluster de bases de données Amazon Neptune gère l'accès à vos données par le biais de requêtes. Un cluster se compose des éléments suivants :

- Une seule instance de base de données principale.
- Jusqu'à 15 instances de base de données de réplica en lecture.

Toutes les instances d'un cluster partagent le même volume de stockage géré sous-jacent, dans le but de garantir fiabilité et haute disponibilité.

Vous vous connectez aux instances de base de données du cluster de bases de données via des points de terminaison Neptune.

Instance de base de données principale dans un cluster de bases de données Neptune

L'instance de base de données principale coordonne toutes les opérations d'écriture sur le volume de stockage sous-jacent du cluster de bases de données. Elle prend également en charge les opérations de lecture.

Il ne peut y avoir qu'une seule instance de base de données principale dans un cluster de bases de données Neptune. Si l'instance principale devient indisponible, Neptune bascule automatiquement vers l'une des instances de réplica en lecture avec une priorité que vous pouvez spécifier.

Read-replica Instances de base de données dans un cluster de base de données Neptune

Une fois que vous avez créé l'instance principale d'un cluster de bases de données, vous pouvez créer jusqu'à 15 réplicas en lecture dans ce cluster afin de prendre en charge les requêtes en lecture seule.

Les instances de base de données de réplica en lecture Neptune sont adaptées à la mise à l'échelle de la capacité de lecture, car elles sont entièrement dédiées aux opérations de lecture sur le volume du cluster. Toutes les opérations d'écriture sont gérées par l'instance principale. Chaque instance de base de données de réplica en lecture possède son propre point de terminaison.

Étant donné que le volume de stockage du cluster est partagé entre toutes les instances d'un cluster, toutes les instances de réplica en lecture renvoient les mêmes données pour les résultats des requêtes avec un retard de réplication très faible. Ce retard est généralement bien inférieur à 100 ms après que l'instance principale a écrit une mise à jour, bien qu'il puisse être un peu plus long lorsque le volume d'opérations d'écriture est très élevé.

Le fait qu'une ou plusieurs instances de réplica en lecture soient disponibles dans différentes zones de disponibilité contribue à accroître la disponibilité, car les réplicas en lecture servent de cibles de basculement pour l'instance principale. En d'autres termes, si l'instance principale échoue, Neptune promeut une instance de réplica en lecture au statut d'instance principale. Lorsque cela se produit, une brève interruption est observée pendant le redémarrage de l'instance promue, interruption au cours de laquelle les demandes de lecture et d'écriture adressées à l'instance principale échouent avec une exception.

En revanche, si votre cluster de base de données n'inclut aucune instance de réplication en lecture, votre cluster de base de données reste indisponible en cas de défaillance de l'instance principale tant qu'elle n'a pas été recréée. Re-creating l'instance principale prend beaucoup plus de temps que la promotion d'une réplique en lecture.

Pour garantir une haute disponibilité, nous vous recommandons de créer une ou plusieurs instances de réplica en lecture ayant la même classe d'instances de base de données que l'instance principale et situées dans des zones de disponibilité différentes de celles de l'instance principale. Consultez Tolérance aux pannes pour un cluster de bases de données Neptune.

À l'aide de la console, vous pouvez créer un Multi-AZ déploiement en spécifiant simplement Multi-AZ lors de la création d'un cluster de bases de données. Si un cluster de base de données se trouve dans une seule zone de disponibilité, vous pouvez en faire un Multi-AZ cluster de base de données en ajoutant une réplique de Neptune dans une autre zone de disponibilité.

> **Note**
>
> Vous ne pouvez pas créer d'instance de réplica en lecture chiffrée pour un cluster de bases de données Neptune non chiffré ni d'instance de réplica en lecture non chiffrée pour un cluster de bases de données Neptune chiffré.

Pour plus d'informations sur la création d'une instance de base de données de réplica en lecture Neptune, consultez Création d'une instance de lecteur Neptune à l'aide de la console.

Dimensionnement des instances de base de données dans un cluster de bases de données Neptune

Dimensionnez les instances du cluster de bases de données Neptune en fonction de vos besoins en termes de CPU et de mémoire. Le nombre de vCPU sur une instance détermine le nombre de threads de requête qui traitent les requêtes entrantes. La quantité de mémoire d'une instance détermine la taille du cache de tampon. Elle est utilisée pour stocker des copies des pages de données extraites du volume de stockage sous-jacent.

Chaque instance de base de données Neptune possède un nombre de threads de requête égal à deux fois le nombre de vCPU sur cette instance. Une instance `r5.4xlarge`, par exemple, dotée de 16 vCPU, possède 32 threads de requêtes et peut donc traiter 32 requêtes simultanément.

Les requêtes supplémentaires qui arrivent alors que tous les threads de requête sont occupés sont placées dans une file d'attente côté serveur et traitées selon le principe du « premier entré, premier sorti » (FIFO) au fur et à mesure que les threads deviennent disponibles. Cette file d'attente côté serveur peut contenir environ 8 000 demandes en attente. Une fois qu'elle est pleine, Neptune répond aux demandes supplémentaires avec une exception `ThrottlingException`. Vous pouvez surveiller le nombre de demandes en attente à l'aide de la `MainRequestQueuePendingRequests` CloudWatch métrique ou en utilisant le point de terminaison d'état des requêtes Gremlin avec le `includeWaiting` paramètre.

Du point de vue du client, le temps d'exécution de la requête inclut le temps passé dans la file d'attente, en plus du temps nécessaire à l'exécution effective de la requête.

Une charge d'écriture simultanée soutenue qui utilise tous les threads de requête de l'instance de base de données principale indique idéalement une utilisation du CPU de 90 % ou plus, ce qui indique que tous les threads de requête du serveur sont activement impliqués dans une tâche utile. Cependant, l'utilisation réelle du CPU est souvent légèrement inférieure, même dans le cas d'une charge d'écriture simultanée prolongée. Cela est généralement dû au fait que les threads de requête attendent la I/O fin des opérations sur le volume de stockage sous-jacent. Neptune utilise des écritures de quorum qui effectuent six copies des données dans trois zones de disponibilité. Quatre de ces six nœuds de stockage doivent accuser réception d'une écriture pour que celles-ci soient considérées comme durables. Lorsqu'un thread de requête attend ce quorum provenant du volume de stockage, il est bloqué, ce qui réduit l'utilisation du CPU.

Si vous avez une charge d'écriture en série où vous effectuez une écriture après l'autre et que vous attendez que la première soit terminée avant de commencer la suivante, l'utilisation du CPU devrait

être encore plus faible. Le montant exact sera fonction du nombre de vCPU et de threads de requête (plus il y a de threads de requêtes, moins le processeur global par requête est élevé), avec une certaine réduction due à l'attente. I/O

Pour plus d'informations sur la meilleure façon de dimensionner les instances de base de données, consultez Choix des types d'instances pour Amazon Neptune. Pour connaître la tarification de chaque type d'instance, consultez la page de tarification de Neptune.

Surveillance des performances des instances de base de données dans Neptune

Vous pouvez utiliser CloudWatch les métriques de Neptune pour surveiller les performances de vos instances de base de données et suivre la latence des requêtes observée par le client. Consultez Utilisation CloudWatch pour surveiller les performances des instances de base de données dans Neptune.

Stockage, fiabilité et disponibilité d'Amazon Neptune

Amazon Neptune utilise une architecture de stockage distribuée et partagée qui se met automatiquement à l'échelle en fonction de vos besoins en stockage de base de données.

Les données Neptune sont stockées dans un volume de cluster, qui est un volume virtuel unique qui utilise des disques Non-Volatile Memory Express (NVMe). SSD-based Le volume du cluster se compose d'un ensemble de blocs logiques, appelés segments. Chacun de ces segments se voit attribuer 10 gigaoctets (Go) de stockage. Les données de chaque segment sont répliquées en six copies, qui sont ensuite distribuées entre trois zones de disponibilité dans la région AWS où se trouve le cluster de bases de données.

Lorsqu'un cluster de bases de données Neptune est créé, un seul segment de 10 Go lui est alloué. Lorsque le volume de données augmente et dépasse le stockage actuellement alloué, Neptune augmente automatiquement le volume du cluster en ajoutant de nouveaux segments. Un volume de cluster Neptune peut atteindre une taille maximale de 128 tebioctets (TiB) dans toutes les régions prises en charge, à l'exception de la Chine GovCloud, où il est limité à 64 TiB. Toutefois, pour les versions du moteur antérieures à Sortie : 1.0.2.2 (09/03/2020), la taille des volumes de cluster est limitée à 64 Tio dans toutes les régions.

Le volume de cluster de bases de données contient toutes les données utilisateur, les index et les dictionnaires (décrits dans la section Modèle de données de graphe de Neptune), ainsi que les métadonnées internes, telles que les journaux de transactions internes. Toutes ces données de graphe, y compris les indicateurs et les journaux internes, ne peuvent pas dépasser la taille maximale du volume de cluster.

I/O—Option de stockage optimisée

Neptune propose deux modèles de tarification pour le stockage :

- Stockage standard : le stockage standard fournit un stockage de base de données économique pour les applications dont l' I/O utilisation est modérée à faible.
- I/O—Stockage optimisé — Avec I/O —Stockage optimisé, vous ne payez que pour le stockage que vous utilisez, à un coût supérieur à celui du stockage standard, et vous ne payez rien pour I/O ce que vous utilisez.

  I/O—Le stockage optimisé est conçu pour répondre aux besoins des charges de travail graphiques I/O intensives à un coût prévisible, avec une faible I/O latence et un débit constant. I/O

Pour plus d'informations, voir I/O—Stockage optimisé.

Allocation du stockage Neptune

Même si un volume de cluster Neptune peut atteindre une taille de 128 Tio (ou 6 Tio dans quelques régions), vous n'êtes facturé que pour l'espace réellement alloué. L'espace total alloué est déterminé par la limite supérieure, qui est la quantité maximale allouée au volume du cluster à tout moment au cours de son existence.

En d'autres termes, même si les données utilisateur sont supprimées d'un volume de cluster, par exemple par le biais d'une requête de suppression comme `g.V().drop()`, l'espace total alloué reste le même. Neptune optimise automatiquement l'espace alloué inutilisé pour pouvoir le réutiliser ultérieurement.

Outre les données utilisateur, deux autres types de contenu consomment de l'espace de stockage interne, à savoir les données du dictionnaire et les journaux de transactions internes. Bien que les données du dictionnaire soient stockées avec des données de graphe, elles persistent indéfiniment, même lorsque les données de graphe prises en charge ont été supprimées, ce qui signifie que les entrées peuvent être réutilisées si les données sont réintroduites. Les données des journaux internes sont stockées dans un espace de stockage interne distinct doté de sa propre limite supérieure. Lorsqu'un journal interne expire, l'espace de stockage qu'il occupait peut être réutilisé pour d'autres journaux, mais pas pour les données de graphe. La quantité d'espace interne allouée aux journaux est incluse dans l'espace total indiqué par la `VolumeBytesUsed` CloudWatch métrique.

Consultez Bonnes pratiques de stockage pour déterminer comment réduire au minimum l'espace de stockage alloué et réutiliser l'espace.

Facturation du stockage Neptune

Les frais de stockage sont facturés sur la base de la limite supérieure, comme décrit ci-dessus. Bien que vos données soient répliquées en six copies, une seule copie des données vous est facturée.

Vous pouvez déterminer le niveau de stockage maximal actuel de votre cluster de base de données en surveillant la `VolumeBytesUsed` CloudWatch métrique (voirSurveillance de Neptune à l'aide d'Amazon CloudWatch).

Parmi les autres facteurs susceptibles d'affecter les coûts de stockage Neptune, citons les instantanés de base de données et les sauvegardes. Ceux-ci sont facturés séparément en tant

que stockage de sauvegarde et sont basés sur les coûts de stockage Neptune (voir CloudWatch métriques utiles pour gérer le stockage de sauvegarde Neptune).

Toutefois, si vous créez un clone de votre base de données, celui-ci pointe vers le même volume de cluster que celui utilisé par votre cluster de bases de données lui-même. Vous n'encourez donc pas de frais de stockage supplémentaires pour les données d'origine. Les modifications ultérieures apportées au clone utilisent le protocole de copie sur écriture et entraînent des coûts de stockage supplémentaires.

Pour plus d'informations sur la tarification de Neptune, consultez Tarification Amazon Neptune.

Bonnes pratiques de stockage Neptune

Étant donné que certains types de données nécessitent un stockage permanent dans Neptune, appliquez les bonnes pratiques suivantes pour éviter des pics importants de croissance du stockage :

- Lorsque vous concevez votre modèle de données de graphe, évitez autant que possible d'utiliser des clés de propriété et des valeurs destinées à l'utilisateur qui sont de nature temporaire.
- Si vous prévoyez d'apporter des modifications à votre modèle de données, ne chargez pas de données sur un cluster de bases de données existant à l'aide du nouveau modèle tant que vous n'avez pas effacé les données de ce cluster de bases de données à l'aide de l'API de réinitialisation rapide. Il est souvent préférable de charger les données utilisant un nouveau modèle sur un nouveau cluster de bases de données.
- Les transactions qui gèrent de grandes quantités de données génèrent des journaux internes d'une taille correspondante, ce qui contribue à augmenter de façon permanente la limite supérieure d'espace des journaux internes. Par exemple, une seule transaction qui supprime toutes les données de votre cluster de bases de données peut générer un énorme journal interne qui nécessiterait l'allocation d'une grande quantité de stockage interne et réduirait ainsi de façon permanente l'espace disponible pour les données de graphe.

  Pour éviter cela, divisez les transactions volumineuses en transactions de plus petite taille et laissez du temps s'écouler entre elles afin que les journaux internes associés aient une chance d'expirer et de libérer leur espace de stockage interne afin qu'il puisse être réutilisé par les journaux suivants.
- Pour surveiller la croissance du volume de votre cluster Neptune, vous pouvez définir une CloudWatch alarme sur la `VolumeBytesUsed` CloudWatch métrique. Cela peut s'avérer particulièrement utile si vos données atteignent la taille maximale du volume de cluster. Pour plus d'informations, consultez la section Utilisation des CloudWatch alarmes Amazon.

La seule façon de réduire l'espace de stockage utilisé par votre cluster de bases de données lorsque vous disposez d'une grande quantité d'espace non utilisé consiste à exporter toutes les données de votre graphe, puis à les recharger dans un nouveau cluster de bases de données. Consultez le service et utilitaire d'exportation de données de Neptune pour exporter facilement des données depuis un cluster de bases de données, et le chargeur en bloc Neptune pour réimporter facilement des données dans Neptune.

> **Note**
>
> La création et la restauration d'un instantané ne réduisent pas la quantité de stockage allouée au cluster de bases de données, car un instantané conserve l'image d'origine du stockage sous-jacent du cluster. Si une quantité substantielle du stockage alloué n'est pas utilisée, la seule façon de la réduire la quantité de stockage alloué est d'exporter toutes les données de votre graphe, puis de les recharger dans un nouveau cluster de bases de données.

Fiabilité et haute disponibilité du stockage Neptune

Amazon Neptune est conçu pour être fiable, durable et tolérant aux pannes.

Le fait que six copies de vos données Neptune soient conservées dans trois zones de disponibilité (AZ) garantit que le stockage des données est extrêmement durable, avec un très faible risque de perte de données. Les données sont répliquées automatiquement dans toutes les zones de disponibilité, qu'elles contiennent ou non des instances de base de données, et le volume de la réplication est indépendant du nombre d'instances de base de données du cluster.

Vous pouvez donc ajouter rapidement un réplica en lecture, car Neptune ne fait pas de nouvelle copie des données du graphe. Au lieu de cela, le réplica en lecture se connecte au volume partagé qui contient déjà toutes les données. De même, la suppression d'un réplica en lecture ne supprime aucune des données sous-jacentes.

Vous ne pouvez supprimer le volume du cluster et ses données qu'après avoir supprimé toutes ses instances de base de données.

De plus, Neptune détecte automatiquement les défaillances au niveau des segments qui composent le volume du cluster. Lorsqu'une copie des données d'un segment est corrompue, Neptune répare immédiatement ce segment en utilisant d'autres copies des données au sein du même segment pour s'assurer que les données réparées sont à jour. En conséquence, Neptune empêche la perte

de données et évite d'avoir à exécuter une restauration à un instant dans le passé pour se remettre d'une défaillance du disque.

Connexion aux points de terminaison Amazon Neptune

Amazon Neptune utilise un cluster d'instances de base de données au lieu d'une seule instance. Chaque connexion Neptune est gérée par une instance de base de données spécifique. Lorsque vous vous connectez à un cluster Neptune, le nom d'hôte et le port que vous spécifiez pointent vers un gestionnaire intermédiaire appelé point de terminaison. Un point de terminaison est une URL qui contient une adresse d'hôte et un port. Les points de terminaison Neptune utilisent des connexions chiffrées de type Transport Security/Secure Layer Sockets Layer ()TLS/SSL.

Neptune utilise le mécanisme du point de terminaison pour abstraire ces connexions, de sorte que vous n'avez pas à coder en dur les noms d'hôtes, ni à écrire votre propre logique de réacheminement des connexions lorsque certaines instances de base de données ne sont pas disponibles.

Les points de terminaison permettent d'associer chaque connexion à l'instance ou au groupe d'instances approprié en fonction de votre cas d'utilisation. Les points de terminaison personnalisés vous permettent de vous connecter à des sous-ensembles d'instances de base de données. Un cluster de bases de données Neptune comprend les points de terminaison suivants :

Points de terminaison de cluster Neptune

Un point de terminaison de cluster est le point de terminaison d'un cluster de bases de données Neptune qui se connecte à l'instance principale de ce cluster. Chaque cluster de bases de données Neptune possède un point de terminaison de cluster et une instance de base de données principale.

Le point de terminaison du cluster fournit un support de basculement pour read/write les connexions au cluster de base de données. Utilisez le point de terminaison de cluster pour toutes les opérations d'écriture sur le cluster DB, y compris les insertions, les mises à jour, les suppression et les modifications de langage de définition de données (DDL). Vous pouvez aussi utiliser le point de terminaison de cluster pour les opérations de lecture, par exemple les requêtes.

En cas de défaillance de l'instance de base de données principale actuelle d'un cluster de bases de données, Neptune bascule automatiquement vers une nouvelle instance de base de données principale. Pendant un basculement, le cluster DB continue à traiter les demandes de connexion adressées au point de terminaison de cluster par la nouvelle instance DB principale, avec une interruption de service minimale.

L'exemple suivant illustre le point de terminaison de cluster d'un cluster de bases de données Neptune.

```
mydbcluster.cluster-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Points de terminaison de lecteur Neptune

Un point de terminaison de lecteur est le point de terminaison d'un cluster de bases de données qui se connecte à l'un des réplicas Neptune disponibles pour ce cluster. Chaque cluster de bases de données Neptune possède un point de terminaison de lecteur. S'il existe plusieurs réplicas, le point de terminaison de lecteur dirige chaque demande de connexion vers l'un des réplicas Neptune.

Le point de terminaison du lecteur assure le routage en tourniquet (round robin) pour les connexions en lecture seule au cluster DB. Utilisez le point de terminaison de lecteur pour les opérations de lecture, par exemple les requêtes .

Vous ne pouvez pas utiliser le point de terminaison du lecteur pour les opérations d'écriture sauf si vous disposez d'un cluster à instance unique (un cluster sans réplicas en lecture). Dans ce cas et dans ce cas seulement, le lecteur peut être utilisé pour les opérations d'écriture ainsi que pour les opérations de lecture.

Le routage en tourniquet du point de terminaison de lecteur modifie l'hôte vers lequel l'entrée DNS pointe. Chaque fois que vous résolvez l'entrée DNS, vous obtenez une autre adresse IP et les connexions sont ouvertes sur ces adresses IP. Une fois qu'une connexion est établie, toutes les demandes relatives à cette connexion sont envoyées au même hôte. Le client doit créer une connexion et résoudre l'enregistrement DNS à nouveau pour obtenir une connexion vers un réplica en lecture potentiellement différent.

> **Note**
>
> WebSockets les connexions sont souvent maintenues en vie pendant de longues périodes. Pour obtenir les différents réplicas en lecture, procédez comme suit :
>
> - Assurez-vous que votre client résout l'entrée DNS chaque fois qu'il se connecte.
> - Fermez la connexion et reconnectez-vous.

Plusieurs clients logiciels peuvent résoudre les entrées DNS de différentes façons. Par exemple, si votre client résout l'entrée DNS, puis utilise l'adresse IP pour chaque connexion, il dirigera toutes les demandes vers un seul hôte.

La mise en cache de l'entrée DNS pour les clients ou proxies sera à même de résoudre le nom DNS pour le même point de terminaison à partir du cache. Il s'agit d'un problème pour les scénarios de basculement et de routage en tourniquet (round-robin).

> **Note**
>
> Désactivez les paramètres de mise en cache DNS pour forcer systématiquement la résolution DNS.

Le cluster de bases de données répartit les demandes de connexion adressées au point de terminaison de lecteur entre les réplicas Neptune disponibles. Si le cluster de bases de données ne contient qu'une instance de base de données principale, le point de terminaison de lecteur traite les demandes de connexion de l'instance de base de données principale. Si un réplica Neptune est créé pour ce cluster de bases de données, le point de terminaison de lecteur continue à traiter les demandes de connexion qui lui sont adressées à partir du nouveau réplica Neptune avec une interruption de service minimale.

L'exemple suivant illustre le point de terminaison de lecteur d'un cluster de bases de données.

```
mydbcluster.cluster-ro-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Points de terminaison d'instance Neptune

Un point de terminaison d'instance est le point de terminaison d'une instance de base de données d'un cluster de bases de données, qui se connecte à cette instance. Chaque instance de base de données d'un cluster de bases de données, quel que soit le type d'instance, a son propre point de terminaison d'instance unique. Par conséquent, il y a un point de terminaison d'instance pour l'instance de base de données principale actuelle du cluster DB. Il existe également un point de terminaison d'instance pour chacun des réplicas Neptune dans le cluster de bases de données.

Le point de terminaison d'instance exerce un contrôle direct sur les connexions au cluster de bases de données, pour les scénarios où l'utilisation du point de terminaison de cluster ou du point de terminaison de lecteur peut ne pas être appropriée. Par exemple, votre application client peut exiger un équilibrage de charge optimisé en fonction de la charge de travail. Dans ce cas, vous pouvez configurer plusieurs clients afin d'obtenir une connexion à différents réplicas Neptune dans un cluster de bases de données pour répartir les charges de travail de lecture.

L'exemple suivant illustre le point de terminaison d'instance de l'une des instances de base de données d'un cluster de bases de données.

```
mydbinstance.123456789012.us-east-1.neptune.amazonaws.com:8182
```

Points de terminaison personnalisés Neptune

Un point de terminaison personnalisé pour un cluster Neptune représente un ensemble d'instances de base de données que vous choisissez. Lorsque vous vous connectez à ce point de terminaison, Neptune choisit l'une des instances dans le groupe pour qu'elle gère la connexion. C'est vous qui définissez les instances auxquelles ce point de terminaison renvoie, ainsi que la fonction même de chaque point de terminaison.

Un cluster de bases de données Neptune ne possède aucun point de terminaison personnalisé tant que vous n'en créez pas un. Vous pouvez créer jusqu'à cinq points de terminaison personnalisés pour chaque cluster Neptune provisionné.

Le point de terminaison personnalisé fournit des connexions de base de données équilibrées en fonction de critères autres que la lecture seule ou la read/write capacité des instances de base de données. Comme la connexion peut exploiter n'importe quelle instance de base de données associée au point de terminaison, assurez-vous que toutes les instances de ce groupe partagent les mêmes caractéristiques de performances et de capacité de mémoire. Lorsque vous avez recours à des points de terminaison personnalisés, vous n'utilisez généralement pas le point de terminaison du lecteur pour ce cluster.

Cette fonctionnalité est destinée aux utilisateurs avancés dotés de types de charges de travail spécialisés où il n'est pas pratique que tous les réplicas Neptune du cluster soient identiques. Les points de terminaison personnalisés vous permettent d'ajuster la capacité des instances de base de données utilisées avec chaque connexion.

Par exemple, si vous définissez plusieurs points de terminaison personnalisés qui se connectent à des groupes d'instances dotés de différentes classes d'instances, vous pouvez diriger les utilisateurs ayant des besoins de performance différents vers les points de terminaison les mieux adaptés à leurs cas d'utilisation.

L'exemple suivant illustre le point de terminaison personnalisé de l'une des instances de base de données d'un cluster de bases de données Neptune.

```
myendpoint.cluster-custom-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Pour plus d'informations, consultez Utilisation des points de terminaison personnalisés.

Considérations relatives aux points de terminaison Neptune

Prenez en compte les points suivants lorsque vous utilisez des points de terminaison Neptune :

- Avant d'utiliser un point de terminaison d'instance pour vous connecter à une instance de base de données spécifique d'un cluster de bases de données, envisagez d'utiliser à la place le point de terminaison de cluster ou de lecteur du cluster de bases de données.

  Le point de terminaison du cluster et le point de terminaison du lecteur assurent la prise en charge des scénarios de haute disponibilité. En cas de défaillance de l'instance de base de données principale d'un cluster de bases de données, Neptune bascule automatiquement vers une nouvelle instance de base de données principale. Pour ce faire, il promeut un réplica Neptune existant en nouvelle instance de base de données principale ou il crée une autre instance de base de données principale. En cas de basculement, vous pouvez soit utiliser le point de terminaison de cluster pour vous reconnecter à l'instance de base de données principale qui vient d'être promue ou créée, soit utiliser le point de terminaison de lecteur pour vous reconnecter à l'un des autres réplicas Neptune du cluster de bases de données.

  Si vous n'adoptez pas cette approche, vous pouvez tout de même vous assurer que vous vous connectez à l'instance de base de données appropriée du cluster de bases de données pour l'opération prévue. Pour ce faire, vous pouvez détecter manuellement ou par programmation l'ensemble obtenu d'instances de base de données disponibles dans le cluster de bases de données et confirmer leurs types d'instance après le basculement, avant d'utiliser le point de terminaison d'instance d'une instance de base de données spécifique.

  Pour plus d'informations sur les basculements, consultez Tolérance aux pannes pour un cluster de bases de données Neptune.

- Le point de terminaison de lecteur dirige uniquement les connexions vers les réplicas Neptune disponibles d'un cluster de bases de données Neptune. Il ne dirige pas les requêtes spécifiques.

  > **Important**
  >
  > Neptune n'équilibre pas la charge.

  Si vous souhaitez équilibrer la charge des requêtes afin de répartir la charge de travail en lecture d'un cluster DB, vous devrez gérer cela dans votre application. Vous devez utiliser les points de terminaison d'instance pour vous connecter directement aux réplicas Neptune afin d'équilibrer la charge.

- Le routage en tourniquet du point de terminaison de lecteur modifie l'hôte vers lequel l'entrée DNS pointe. Le client doit créer une connexion et résoudre l'enregistrement DNS à nouveau pour obtenir une connexion vers un potentiel nouveau réplica en lecture.

- Pendant un basculement, le point de terminaison de lecteur peut brièvement diriger les connexions vers la nouvelle instance principale d'un cluster de bases de données, lorsqu'un réplica Neptune est promu en nouvelle instance de base de données principale.

Utilisation des points de terminaison personnalisés dans Neptune

Lorsque vous ajoutez ou supprimez une instance de base de données dans un point de terminaison personnalisé, toutes les connexions existantes à cette instance restent actives.

Vous pouvez définir une liste d'instances de base de données à inclure dans un point de terminaison personnalisé (liste statique) ou une liste d'instances de base de données à exclure du point de terminaison personnalisé (liste d'exclusion). Vous pouvez utiliser le inclusion/exclusion mécanisme pour subdiviser les instances de base de données en groupes et vous assurer que les points de terminaison personnalisés couvrent toutes les instances de base de données du cluster. Chaque point de terminaison personnalisé ne peut contenir qu’un de ces types de liste.

Dans le AWS Management Console, le choix est représenté par la case à cocher Attacher les futures instances ajoutées à ce cluster. Lorsque cette case n'est pas cochée, le point de terminaison personnalisé utilise une liste statique contenant uniquement les instances de base de données spécifiées dans la boîte de dialogue. Lorsque vous cochez cette case, le point de terminaison personnalisé utilise une liste d'exclusion. Dans ce cas, le point de terminaison personnalisé représente toutes les instances de base de données du cluster (y compris celles que vous ajouterez par la suite), à l'exception de celles qui ne sont pas sélectionnées dans la boîte de dialogue.

Neptune ne modifie pas les instances de base de données spécifiées dans les listes statiques ou d'exclusion lorsqu'elles passent du rôle d'instance principale et au rôle de réplica Neptune, ou inversement, en raison d'un basculement ou d'une opération de promotion.

Vous pouvez associer une instance de base de données à plusieurs points de terminaison personnalisés. Supposons, par exemple, que vous ajoutiez une nouvelle instance de base de données à un cluster. Dans ce cas, l'instance de base de données est ajoutée à tous les points de

terminaison personnalisés auxquels elle est éligible. La liste statique ou d'exclusion définie pour ce cluster détermine quelle instance de base de données peut y être ajoutée.

Si le point de terminaison comprend une liste statique d'instances de base de données, les réplicas Neptune qui viennent d'être ajoutés ne sont pas inclus dans ce point de terminaison. Inversement, si le point de terminaison dispose d'une liste d'exclusion, les réplicas Neptune qui viennent d'être ajoutés y sont inclus s'ils ne font pas partie de la liste d'exclusion.

Si un réplica Neptune devient indisponible, il reste associé aux points de terminaison personnalisés. Cette association persiste qu'il ne soit pas sain, qu'il soit arrêté, qu'il en cours de redémarrage ou qu'il ne soit pas disponible pour une autre raison. Toutefois, tant qu'il n'est pas disponible, vous ne pouvez vous y connecter via un point de terminaison.

Comme les nouveaux clusters Neptune ne contiennent pas de points de terminaison personnalisés, vous devez les créer et les gérer vous-même. Cela est également vrai pour les clusters Neptune restaurés à partir d'instantanés, car les points de terminaison personnalisés ne sont pas inclus dans l'instantané. Vous devrez les créer à nouveau après leur restauration, et de nouveaux noms de points de terminaison seront choisis si le cluster restauré se trouve dans la même région que celui d'origine.

Création d'un point de terminaison personnalisé

Gérez les points de terminaison personnalisés à l'aide de la console Neptune. Pour ce faire, accédez à la page de détails du cluster Neptune et utilisez les commandes de la section Points de terminaison personnalisés.

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Accédez à la page de détails du cluster.
3. Choisissez l'action `Create custom endpoint` dans la section Points de terminaison.
4. Choisissez un nom pour le point de terminaison personnalisé. Ce nom sera spécifique à votre ID utilisateur et à votre région. Il doit comporter 63 caractères maximum et prendre la forme suivante :

   endpointName`.cluster-custom-`*customerDnsIdentifier*`.`*dnsSuffix*

   Comme le nom des points de terminaison personnalisés ne comprend pas le nom du cluster, vous n'avez pas à le mettre à jour en cas de modification du nom de ce cluster. Cependant, vous ne pouvez pas réutiliser le même nom de point de terminaison personnalisé pour d'autres clusters de la même région. Donnez à chaque point de terminaison personnalisé un nom unique parmi tous les clusters affectés à votre ID utilisateur au sein d'une région spécifique.

5. Pour choisir une liste d'instances de base de données qui restera la même à mesure que le cluster s'élargira, laissez la case Attach future instances added to this cluster (Attacher les instances futures ajoutées à ce cluster) décochée. Lorsque vous cochez cette case, le point de terminaison personnalisé ajoute les nouvelles instances de manière dynamique à mesure qu'elles sont ajoutées au cluster.

Affichage des points de terminaison personnalisés

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Accédez à la page de détails de votre cluster de bases de données.
3. La section Points de terminaison contient uniquement des informations sur les points de terminaison personnalisés (les détails sur les points de terminaison intégrés sont répertoriés dans la section Détails principale). Pour consulter les détails relatifs à un point de terminaison personnalisé, sélectionnez son nom afin d'afficher la page de détails correspondante.

Modification d'un point de terminaison personnalisé

Vous pouvez modifier les propriétés d'un point de terminaison personnalisé pour changer les instances de base de données qui lui sont associées. Vous pouvez également passer d'une liste statique à une liste d'exclusion, ou vice versa.

Il n'est pas possible d'utiliser un point de terminaison personnalisé ou de s'y connecter lorsque les modifications d'une action d'édition sont en cours. Après une modification, plusieurs minutes peuvent être nécessaires avant que le statut du point de terminaison redevienne disponible et que vous puissiez vous y connecter à nouveau.

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Accédez à la page de détails du cluster.
3. Dans la section Points de terminaison), sélectionnez le nom du point de terminaison personnalisé que vous voulez modifier.
4. Sur la page de détails de ce point de terminaison, choisissez l'action Modifier.

Suppression d'un point de terminaison personnalisé

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Accédez à la page de détails du cluster.
3. Dans la section Points de terminaison), sélectionnez le nom du point de terminaison personnalisé que vous voulez supprimer.
4. Sur la page de détails de ce point de terminaison, choisissez l'action Supprimer.

Mode expérimental Neptune

Vous pouvez utiliser le mode laboratoire d'Amazon Neptune pour activer les nouvelles fonctionnalités présentes dans la version actuelle du moteur Neptune, mais qui ne sont pas encore prêtes à être utilisées en production et ne sont pas activées par défaut. Cela vous permet de tester ces fonctionnalités dans vos environnements de développement et de test.

Utilisation du mode expérimental Neptune

Utilisez le paramètre de cluster de bases de données `neptune_lab_mode` pour activer ou désactiver des fonctionnalités. Pour ce faire, vous devez inclure *`(feature name)`*`=enabled` ou *`(feature name)`*`=disabled` dans la valeur du paramètre `neptune_lab_mode` dans le groupe de paramètres de cluster de bases de données.

Par exemple, dans cette version de moteur, vous pouvez définir le paramètre `neptune_lab_mode` sur `Streams=disabled, ReadWriteConflictDetection=enabled`.

Pour plus d'informations sur la modification du groupe de paramètres de cluster de bases de données pour votre base de données, consultez Modification d'un groupe de paramètres. Notez que vous ne pouvez pas modifier le groupe de paramètres de cluster de bases de données par défaut. Si vous utilisez le groupe par défaut, vous devez créer un groupe de paramètres de cluster de bases de données pour pouvoir définir le paramètre `neptune_lab_mode`.

> **Note**
>
> Lorsque vous modifiez un paramètre de cluster de bases de données statique (par exemple, `neptune_lab_mode`), vous devez redémarrer l'instance principale (enregistreur) du cluster pour que la modification prenne effet. Avant la Sortie : 1.2.0.0 (21/07/2022), tous les réplicas en lecture d'un cluster de bases de données étaient automatiquement redémarrés lors du redémarrage de l'instance principale.
> À compter de la Sortie : 1.2.0.0 (21/07/2022), le redémarrage de l'instance principale n'entraîne le redémarrage d'aucun des réplicas. En d'autres termes, vous devez redémarrer chaque instance séparément pour récupérer une modification des paramètres du cluster de bases de données (voir Groupes de paramètres).

Important

À l'heure actuelle, si vous fournissez de mauvais paramètres en mode expérimental ou si votre demande échoue pour une autre raison, il se peut que vous ne soyez pas informé de l'échec. Vous devez toujours vérifier qu'une demande de changement en mode expérimental a abouti en appelant l'API de statut comme indiqué ci-dessous :

```
curl -G https://your-neptune-endpoint:port/status
```

Les résultats de statut incluent des informations relatives au mode expérimental. Elles indiquent si les modifications que vous avez demandées ont été apportées ou non :

```
{
  "status":"healthy",
  "startTime":"Wed Dec 29 02:29:24 UTC 2021",
  "dbEngineVersion":"development",
  "role":"writer",
  "dfeQueryEngine":"viaQueryHint",
  "gremlin":{"version":"tinkerpop-3.5.2"},
  "sparql":{"version":"sparql-1.1"},
  "opencypher":{"version":"Neptune-9.0.20190305-1.0"},
  "labMode":{
    "ObjectIndex":"disabled",
    "ReadWriteConflictDetection":"enabled"
  },
  "features":{
    "LookupCache":{"status":"Available"},
    "ResultCache":{"status":"disabled"},
    "IAMAuthentication":"disabled",
    "Streams":"disabled",
    "AuditLog":"disabled"
  },
  "settings":{"clusterQueryTimeoutInMs":"120000"}
}
```

Les fonctionnalités suivantes sont actuellement accessibles en mode laboratoire :

Index OSGP

Neptune peut désormais gérer un quatrième index, à savoir l'index OSGP, qui s'avère utile pour les jeux de données comportant un grand nombre de prédicats (voir Activation d'un index OSGP).

Vous pouvez activer un index OSGP dans un nouveau cluster de bases de données Neptune vide en définissant `ObjectIndex=enabled` dans le paramètre de cluster de bases de données `neptune_lab_mode`. Un index OSGP ne peut être activé que dans un nouveau cluster de bases de données vide.

Par défaut, l'index OSGP est désactivé.

> **Note**
>
> Après avoir défini le paramètre du cluster de bases de données `neptune_lab_mode` afin d'activer l'index OSGP, vous devez redémarrer l'instance d'enregistreur du cluster pour que la modification prenne effet.

> **Warning**
>
> Si vous désactivez un index OSGP activé en le définissant `ObjectIndex=disabled` puis en le réactivant ultérieurement après avoir ajouté des données supplémentaires, l'index ne sera pas créé correctement. On-demand la reconstruction de l'index n'étant pas prise en charge, vous ne devez activer l'index OSGP que lorsque la base de données est vide.

Activer la collecte des déchets dans les dictionnaires

La collecte des déchets du dictionnaire peut être activée pour les données du graphe de propriétés lorsque neptune-streams n'est pas activé via le paramètre. `DictionaryGCMode` La simultanéité peut être contrôlée via le `DictionaryGCConcurrency` paramètre. Pour plus d'informations, consultez Dictionnaire de collecte des déchets.

Sémantique des transactions formalisée

Neptune a mis à jour la sémantique formelle des transactions simultanées (voir Sémantique des transactions dans Neptune).

Utilisez `ReadWriteConflictDetection` comme nom dans le paramètre `neptune_lab_mode` qui active ou désactive la sémantique de transaction formalisée.

Par défaut, la sémantique des transactions formalisée est déjà activée. Si vous souhaitez revenir au comportement antérieur, incluez `ReadWriteConflictDetection=disabled` dans la valeur définie pour le paramètre `neptune_lab_mode` du cluster de bases de données.

Support date/heure étendu

Neptune a étendu la prise en charge de la fonctionnalité datetime. Pour activer la date et l'heure avec des formats étendus, `DatetimeMillisecond=enabled` incluez-le dans la valeur définie pour le `neptune_lab_mode` paramètre DB Cluster.

StrictTimeoutValidation

> **Note**
>
> Cette fonctionnalité est disponible à partir de la version 1.3.2.0 du moteur Neptune.

Valeur par défaut : activé (désactivé par défaut avant la version 1.4.0.0 du moteur Neptune)

Valeurs autorisées : enabled/disabled

Lorsque ce paramètre est définie`enabled`, une valeur de délai par requête spécifiée sous forme d'option de demande ou d'indice de requête ne peut pas dépasser la valeur définie globalement dans le réglage du groupe de `neptune_query_timeout`paramètres. Si le délai d'expiration par requête dépasse le paramètre global, Neptune lance un. `InvalidParameterException` Dans les versions du moteur antérieures à la version 1.4.0.0, ce paramètre était `disabled` par défaut et devait être explicitement activé.

Ce paramètre peut être confirmé dans une réponse sur le `/status` terminal lorsque la valeur est`disabled`.

Pour de plus amples informations, veuillez consulter Délais d'expiration au niveau de chaque requête.

AccurateQRCMemoryEstimation

> **Note**
>
> Cette fonctionnalité est disponible à partir de la version 1.4.0.0 du moteur Neptune.

Valeur par défaut : désactivé

Valeurs autorisées : enabled/disabled

Lorsqu'il est activé, le cache des résultats des requêtes Gremlin permet de mettre en cache les résultats des requêtes dans la base de données. Par défaut, l'estimation approximative est utilisée pour déterminer la taille du résultat mis en cache. Lorsque ce paramètre du mode laboratoire est `AccurateQRCMemoryEstimation` activé, l'estimation de la taille des résultats mis en cache utilisera des estimations de taille précises plutôt qu'approximatives. Ce paramètre labmode est disponible à partir de la version 1.4.0.0 du moteur Neptune.

Autre moteur de requête (DFE) Amazon Neptune

Amazon Neptune dispose d'un moteur de requête alternatif appelé DFE qui utilise les ressources des instances de base de données telles que les cœurs de processeur, la mémoire, et ce de manière I/O plus efficace que le moteur Neptune d'origine.

> **Note**
>
> Avec les jeux de données volumineux, le moteur DFE peut ne pas fonctionner correctement sur les instances t3.

Le moteur DFE exécute les requêtes SPARQL, Gremlin et openCypher, et prend en charge une grande variété de types de plans, y compris les arbres linéaires gauches, les arbres « bushy » et les plans hybrides. Les opérateurs de plan peuvent invoquer à la fois des opérations de calcul, qui s'exécutent sur un ensemble réservé de cœurs de calcul, et des I/O opérations, chacune s'exécutant sur son propre thread dans un pool de I/O threads.

Le DFE utilise des statistiques prégénérées sur les données de votre graphe Neptune pour prendre des décisions éclairées sur la manière de structurer les requêtes. Consultez Statistiques DFE pour en savoir plus sur la façon dont ces statistiques sont générées.

Le choix du type de plan et du nombre de threads de calcul utilisés est effectué automatiquement en fonction des statistiques prégénérées et des ressources disponibles dans le nœud de tête Neptune. L'ordre des résultats n'est pas prédéterminé pour les plans dotés d'un parallélisme de calcul interne.

Contrôle de l'endroit où le moteur DFE Neptune est utilisé

Par défaut, le paramètre neptune_dfe_query_engine d'une instance est défini sur `viaQueryHint`, ce qui signifie que le moteur DFE n'est utilisé que pour les requêtes openCypher et pour les requêtes Gremlin et SPARQL qui incluent explicitement l'indicateur de requête `useDFE` défini sur `true`.

Vous pouvez activer complètement le moteur DFE afin qu'il soit utilisé autant que possible en définissant le paramètre d'instance `neptune_dfe_query_engine` sur `enabled`.

Vous pouvez également désactiver le DFE en incluant l'indicateur de requête `useDFE` pour une requête Gremlin ou une requête SPARQL particulière. Cet indicateur de requête vous permet d'empêcher le DFE d'exécuter cette requête particulière.

Vous pouvez déterminer si le DFE est activé ou non dans une instance à l'aide d'un appel Statut d'une instance, comme ceci :

```
curl -G https://your-neptune-endpoint:port/status
```

La réponse de statut indique ensuite si le DFE est activé ou non :

```
{
  "status":"healthy",
  "startTime":"Wed Dec 29 02:29:24 UTC 2021",
  "dbEngineVersion":"development",
  "role":"writer",
  "dfeQueryEngine":"viaQueryHint",
  "gremlin":{"version":"tinkerpop-3.5.2"},
  "sparql":{"version":"sparql-1.1"},
  "opencypher":{"version":"Neptune-9.0.20190305-1.0"},
  "labMode":{
    "ObjectIndex":"disabled",
    "ReadWriteConflictDetection":"enabled"
  },
  "features":{
    "ResultCache":{"status":"disabled"},
    "IAMAuthentication":"disabled",
    "Streams":"disabled",
    "AuditLog":"disabled"
  },
  "settings":{"clusterQueryTimeoutInMs":"120000"}
}
```

Les résultats Gremlin `explain` et `profile` indiquent si une requête est exécutée par le DFE. Voir Informations contenues dans un rapport d'`explication` de Gremlin pour `explain` et Rapports `profile` avec le DFE activé pour`profile`.

De même, SPARQL `explain` vous indique si une requête SPARQL est exécutée par le DFE. Pour plus d'informations, consultez Exemple de SPARQL `expliquant` la sortie lorsque le DFE est activé et `opérateur DFEnode`.

Constructions de requête prises en charge par le DFE Neptune

À l'heure actuelle, le DFE Neptune prend en charge un sous-ensemble de constructions de requêtes SPARQL et Gremlin.

Pour SPARQL, il s'agit du sous-ensemble des modèles de graphes de base conjonctifs.

Pour Gremlin, il s'agit généralement du sous-ensemble de requêtes contenant une chaîne de traversées qui ne comporte pas certaines des étapes les plus complexes.

Pour savoir si l'une de vos requêtes est exécutée dans sa totalité ou en partie par le DFE, procédez comme suit :

- Dans Gremlin, les résultats `explain` et `profile` indiquent quelles parties d'une requête sont exécutées par le DFE, le cas échéant. Voir Informations contenues dans un rapport d'`explication` de Gremlin pour `explain` et Rapports `profile` avec le DFE activé pour`profile`. Consultez aussi `Réglage des requêtes Gremlin à l'aide de la fonction Explique et du profil`.

  Les détails relatifs à la prise en charge du moteur Neptune pour les différentes étapes Gremlin sont documentés dans Prise en charge des étapes Gremlin.
- De même, SPARQL `explain` vous indique si une requête SPARQL est exécutée par le DFE. Pour plus d'informations, consultez Exemple de SPARQL `expliquant` la sortie lorsque le DFE est activé et `opérateur DFEnode`.

Gestion des statistiques à utiliser par le DFE Neptune

> **Note**
>
> La prise en charge d'openCypher dépend du moteur de requête DFE de Neptune.
> Le moteur DFE est contrôlé à l'aide du paramètre d'neptune_dfe_query_engineinstance dans le groupe de paramètres de base de données d'une instance.

Le moteur DFE utilise les informations relatives aux données d'un graphe Neptune pour effectuer des compromis efficaces lors de la planification de l'exécution des requêtes. Ces informations prennent la forme de statistiques qui incluent ce que l'on appelle des ensembles de caractéristiques et des statistiques de prédicats qui contribuent à guider la planification des requêtes.

À partir de la version 1.2.1.0 du moteur, vous pouvez récupérer des informations récapitulatives sur votre graphique à partir de ces statistiques à l'aide de l'GetGraphSummaryAPI ou du `summary` point de terminaison.

Ces statistiques DFE sont générées chaque fois que plus de 10 % des données de votre graphe ont changé ou lorsque les dernières statistiques datent de plus de 10 jours. Cependant, ces déclencheurs pourraient changer à l'avenir.

> **Note**
>
> La génération de statistiques est désactivée sur les instances `T3` et `T4g`, car elle peut dépasser la capacité de mémoire de ces types d'instances.

Vous pouvez gérer la génération de statistiques DFE via l'un des points de terminaison suivants :

- `https://`*`your-neptune-host`*`:`*`port`*`/rdf/statistics` (pour SPARQL).
- `https://`*`your-neptune-host`*`:`*`port`*`/propertygraph/statistics` (pour Gremlin et openCypher) et sa version alternative : `https://`*`your-neptune-host`*`:`*`port`*`/pg/statistics`.

> **Note**
>
> Depuis la version 1.1.1.0 du moteur, le point de terminaison de statistiques Gremlin (`https://your-neptune-host:port/gremlin/statistics`) est obsolète et a été remplacé par le point de terminaison `propertygraph` ou `pg`. Il reste pris en charge pour des raisons de rétrocompatibilité, mais il pourrait être supprimé dans les futures versions.
> Depuis la version 1.2.1.0 du moteur, le point de terminaison de statistiques SPARQL (`https://your-neptune-host:port/sparql/statistics`) est obsolète et a été remplacé par le point de terminaison `rdf`. Il reste pris en charge pour des raisons de rétrocompatibilité, mais il pourrait être supprimé dans les futures versions.

> **Note**
>
> Si un point de terminaison de statistiques DFE se trouve sur une instance de lecteur, les seules demandes qu'il peut traiter sont les demandes de statut. Les autres demandes échoueront avec une exception `ReadOnlyViolationException`.

Limites de taille pour la génération des statistiques DFE

Actuellement, la génération des statistiques DFE s'arrête si l'une des limites de taille suivantes est atteinte :

- Le nombre d'ensembles de caractéristiques générés ne doit pas dépasser 50 000.
- Le nombre de statistiques de prédicat générées ne doit pas dépasser un million.

Ces limites sont susceptibles de changer.

Statut actuel des statistiques du DFE

Vous pouvez vérifier l'état actuel des statistiques DFE comme suit :

> **Note**
>
> Les exemples suivants utilisent le point de terminaison et les commandes du graphe de propriétés. Pour les données RDF, utilisez-les `get-sparql-statistics` pour les points de terminaison REST et `/rdf/statistics` pour les points de terminaison. AWS CLI

AWS CLI

```
aws neptunedata get-propertygraph-statistics \
  --endpoint-url https://your-neptune-endpoint:port \
  --region us-east-1
```

> **Note**
>
> Remplacez *us-east-1* par la région de votre cluster Neptune. Vous pouvez omettre `--region` si vous avez configuré une région par défaut.

Pour plus d'informations, consultez get-propertygraph-statistics dans la référence des commandes. AWS CLI

SDK

```
import boto3
import json
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.get_propertygraph_statistics()

print(json.dumps(response['payload'], indent=2, default=str))
```

awscurl

```
awscurl https://your-neptune-endpoint:port/propertygraph/statistics \
```

```
  --region us-east-1 \
  --service neptune-db
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultezUtiliser awscurl avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

curl

```
curl -G https://your-neptune-endpoint:port/propertygraph/statistics
```

La réponse à une demande de statut comporte les champs suivants :

- `status` : code de retour HTTP de la demande. Si la demande aboutit, le code est `200`. Pour obtenir une liste des erreurs courantes, consultez Erreurs courantes.
- `payload`:
  - `autoCompute` : (valeur booléenne) indique si la génération automatique des statistiques est activée ou non.
  - `active` : (valeur booléenne) indique si la génération des statistiques est activée.
  - `statisticsId` : indique l'ID de la génération de statistiques en cours d'exécution. La valeur `-1` indique qu'aucune statistique n'a été générée.
  - `date` : heure UTC à laquelle les dernières statistiques DFE ont été générées, au format ISO 8601.

    > **Note**
    >
    > Avant la version 1.2.1.0 du moteur, elle était représentée avec une précision à la minute, mais depuis la version 1.2.1.0 du moteur, elle est représentée avec une précision à la milliseconde (par exemple, `2023-01-24T00:47:43.319Z`).

  - `note` : remarque concernant les problèmes liés à la non-validité des statistiques.

- `signatureInfo` : contient des informations sur les ensembles de caractéristiques générés dans les statistiques (avant la version 1.2.1.0 du moteur, ce champ était nommé `summary`). Elles ne sont généralement pas directement exploitables :
  - `signatureCount` : nombre total de signatures pour tous les ensembles de caractéristiques.
  - `instanceCount` : nombre total d'instances d'ensembles de caractéristiques.
  - `predicateCount` : nombre total de prédicats uniques.

La réponse à une demande de statut quand aucune statistique n'a été générée ressemble à ceci :

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : -1
   }
}
```

Si des statistiques DFE sont disponibles, la réponse ressemble à ceci :

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : true,
    "statisticsId" : 1588893232718,
    "date" : "2020-05-07T23:13Z",
    "summary" : {
      "signatureCount" : 5,
      "instanceCount" : 1000,
      "predicateCount" : 20
    }
  }
}
```

Si la génération des statistiques DFE a échoué, par exemple parce qu'elle a dépassé la limite de taille des statistiques, la réponse ressemble à ceci :

```
{
```

```
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : 1588713528304,
    "date" : "2020-05-05T21:18Z",
    "note" : "Limit reached: Statistics are not available"
  }
}
```

Désactivation de la génération automatique des statistiques DFE

Par défaut, la génération automatique des statistiques DFE est activée lorsque vous activez le DFE.

Vous pouvez désactiver la génération automatique comme suit :

> **Note**
>
> Les exemples suivants utilisent le point de terminaison et les commandes du graphe de propriétés. Pour les données RDF, utilisez-les `manage-sparql-statistics` pour les points de terminaison REST et `/rdf/statistics` pour les points de terminaison. AWS CLI

AWS CLI

```
aws neptunedata manage-propertygraph-statistics \
  --endpoint-url https://your-neptune-endpoint:port \
  --mode disableAutoCompute
```

Pour plus d'informations, consultez la section manage-propertygraph-statistics dans le manuel de référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)
```

```
response = client.manage_propertygraph_statistics(mode='disableAutoCompute')

print(response['status'])
```

awscurl

```
awscurl https://your-neptune-endpoint:port/propertygraph/statistics \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{"mode":"disableAutoCompute"}'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultezUtiliser awscurl avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

curl

```
curl -X POST -d '{ "mode" : "disableAutoCompute" }' https://your-neptune-
endpoint:port/propertygraph/statistics
```

Si la demande aboutit, voice ce à quoi correspondent le code de réponse HTTP `200` et la réponse :

```
{
  "status" : "200 OK"
}
```

Pour confirmer que la génération automatique est désactivée, émettez une demande de statut et vérifiez que le champ `autoCompute` de la réponse est défini sur `false`.

La désactivation de la génération automatique des statistiques ne met pas fin à un calcul de statistiques en cours.

Si vous demandez de désactiver la génération automatique vers une instance de lecteur plutôt que vers l'instance d'enregistreur de votre cluster de bases de données, la demande échoue avec le code de retour HTTP 400 et génère la sortie suivante :

```
{
  "detailedMessage" : "Writes are not permitted on a read replica instance",
  "code" : "ReadOnlyViolationException",
  "requestId":"8eb8d3e5-0996-4a1b-616a-74e0ec32d5f7"
}
```

Pour obtenir une liste des autres erreurs courantes, consultez Erreurs courantes.

Re-enabling génération automatique de statistiques DFE

Par défaut, la génération automatique des statistiques DFE est déjà activée lorsque vous activez le DFE. Si vous désactivez la génération automatique, vous pouvez la réactiver ultérieurement comme suit :

> **Note**
>
> Les exemples suivants utilisent le point de terminaison et les commandes du graphe de propriétés. Pour les données RDF, utilisez-les `manage-sparql-statistics` pour les points de terminaison REST et `/rdf/statistics` pour les points de terminaison. AWS CLI

AWS CLI

```
aws neptunedata manage-propertygraph-statistics \
  --endpoint-url https://your-neptune-endpoint:port \
  --mode enableAutoCompute
```

Pour plus d'informations, consultez la section manage-propertygraph-statistics dans le manuel de référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config
```

```
client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.manage_propertygraph_statistics(mode='enableAutoCompute')

print(response['status'])
```

awscurl

```
awscurl https://your-neptune-endpoint:port/propertygraph/statistics \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{"mode":"enableAutoCompute"}'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultezUtiliser awscurl avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

curl

```
curl -X POST -d '{ "mode" : "enableAutoCompute" }' https://your-neptune-
endpoint:port/propertygraph/statistics
```

Si la demande aboutit, voice ce à quoi correspondent le code de réponse HTTP `200` et la réponse :

```
{
  "status" : "200 OK"
}
```

Pour confirmer que la génération automatique est activée, émettez une demande de statut et vérifiez que le champ `autoCompute` de la réponse est défini sur `true`.

Déclenchement manuel de la génération de statistiques DFE

Vous pouvez lancer la génération de statistiques DFE manuellement comme suit :

> **Note**
>
> Les exemples suivants utilisent le point de terminaison et les commandes du graphe de propriétés. Pour les données RDF, utilisez-les `manage-sparql-statistics` pour les points de terminaison REST et `/rdf/statistics` pour les points de terminaison. AWS CLI

AWS CLI

```
aws neptunedata manage-propertygraph-statistics \
  --endpoint-url https://your-neptune-endpoint:port \
  --mode refresh
```

Pour plus d'informations, consultez la section manage-propertygraph-statistics dans le manuel de référence des commandes. AWS CLI

SDK

```
import boto3
import json
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.manage_propertygraph_statistics(mode='refresh')

print(json.dumps(response['payload'], indent=2, default=str))
```

awscurl

```
awscurl https://your-neptune-endpoint:port/propertygraph/statistics \
```

```
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{"mode":"refresh"}'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultezUtiliser awscurl avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

curl

```
curl -X POST -d '{ "mode" : "refresh" }' https://your-neptune-endpoint:port/
propertygraph/statistics
```

Si la requête aboutit, le résultat est le suivant, avec le code de retour HTTP 200 :

```
{
  "status" : "200 OK",
  "payload" : {
    "statisticsId" : 1588893232718
  }
}
```

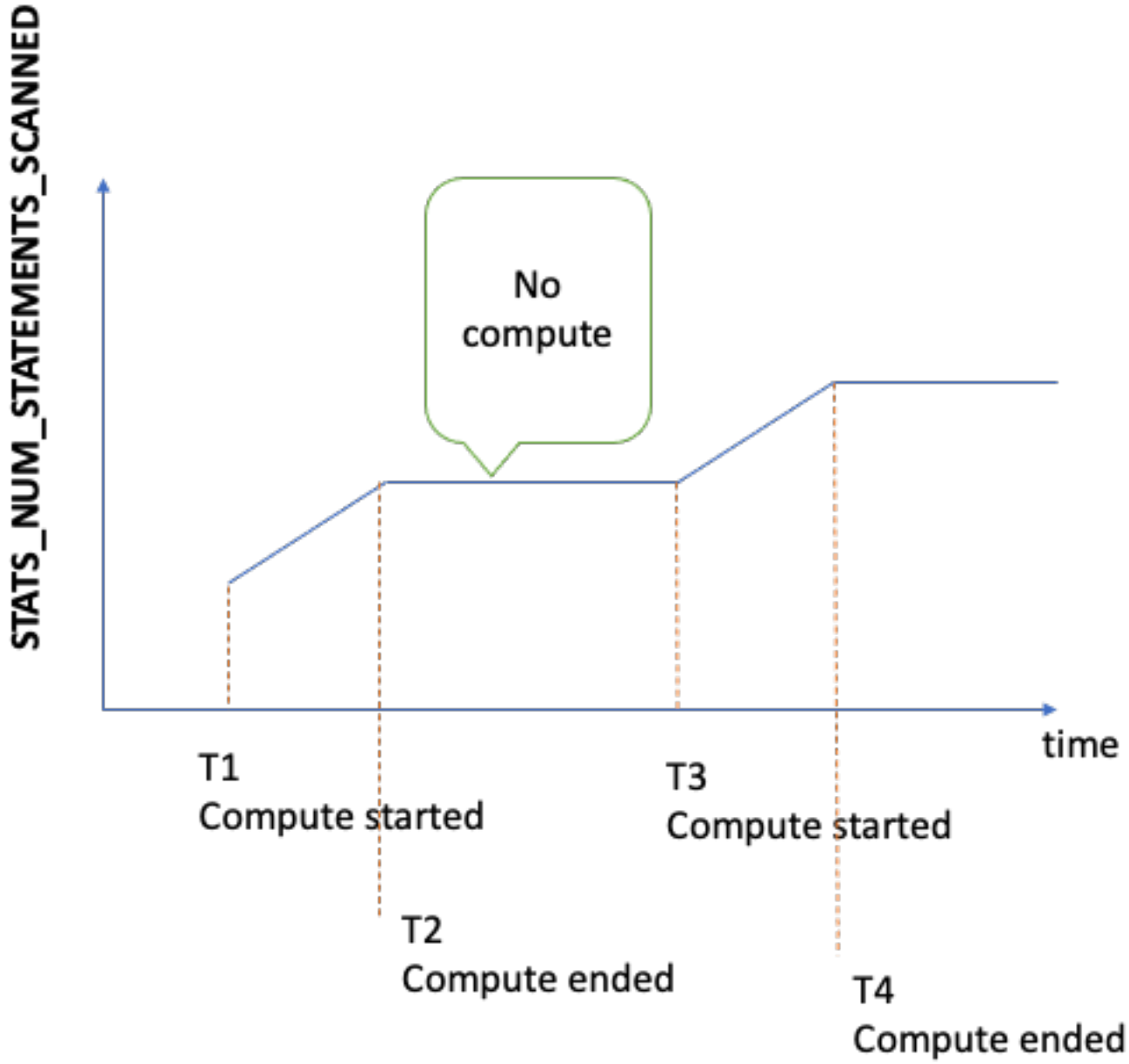
Le `statisticsId` dans la sortie contient l'ID de l'exécution de la génération de statistiques en cours. Si une exécution était déjà en cours au moment de la demande, cette demande renvoie l'ID de cette exécution plutôt que d'en lancer une nouvelle. Une seule génération de statistiques peut avoir lieu à la fois.

En cas de basculement lors de la génération des statistiques DFE, le nouveau nœud d'enregistreur relève le dernier point de contrôle traité et reprend l'exécution des statistiques à partir de là.

Utilisation de la **`StatsNumStatementsScanned`** CloudWatch métrique pour surveiller le calcul des statistiques

La `StatsNumStatementsScanned` CloudWatch métrique renvoie le nombre total d'instructions analysées pour le calcul des statistiques depuis le démarrage du serveur. Elle est mise à jour à chaque tranche de calcul des statistiques.

Chaque fois que le calcul des statistiques est déclenché, ce nombre augmente. Quand aucun calcul n'est effectué, il reste constant. L'examen d'un diagramme de valeurs `StatsNumStatementsScanned` au fil du temps donne donc une idée assez claire de la date et de la rapidité du calcul des statistiques :



Lorsque le calcul est en cours, la pente du graphe indique la vitesse (plus la pente est raide, plus les statistiques sont calculées rapidement).

Si le graphe est simplement une ligne plate à 0, la fonctionnalité de statistiques a été activée, mais aucune statistique n'a été calculée. Si la fonctionnalité de statistiques a été désactivée ou si vous utilisez une version du moteur qui ne prend pas en charge le calcul des statistiques, `StatsNumStatementsScanned` n'existe pas.

Comme indiqué précédemment, vous pouvez désactiver le calcul des statistiques à l'aide de l'API de statistiques. Toutefois, la désactivation de cette fonctionnalité peut empêcher la mise à jour des

statistiques, ce qui peut entraîner une génération inappropriée du plan de requête pour le moteur DFE.

Surveillance de Neptune à l'aide d'Amazon CloudWatchReportez-vous à la section pour plus d'informations sur son utilisation CloudWatch.

Utilisation Gestion des identités et des accès AWS Authentification (IAM) avec les points de terminaison statistiques DFE

Vous pouvez accéder aux points de terminaison des statistiques DFE en toute sécurité grâce à l'authentification IAM à l' AWS CLI aide de awscurl ou de tout autre outil compatible avec HTTPS et IAM. Pour plus d'informations sur la configuration des informations d'identification appropriées, consultezUtiliser `awscurl` avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

Les exemples suivants montrent comment effectuer une demande de statut authentifiée :

> **Note**
>
> Les exemples suivants utilisent le point de terminaison et les commandes du graphe de propriétés. Pour les données RDF, utilisez les commandes SPARQL équivalentes (`get-sparql-statistics`,`manage-sparql-statistics`) pour les points de terminaison REST et `/rdf/statistics` pour AWS CLI les points de terminaison.

AWS CLI

```
aws neptunedata get-propertygraph-statistics \
  --endpoint-url https://your-neptune-endpoint:port
```

Gère automatiquement AWS CLI l'authentification IAM à l'aide de vos informations d'identification configurées.

SDK

```
import boto3
import json
from botocore.config import Config

client = boto3.client(
```

```
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.get_propertygraph_statistics()

print(json.dumps(response['payload'], indent=2, default=str))
```

Le SDK gère automatiquement l'authentification IAM à l'aide des informations d'identification que vous avez configurées.

awscurl

```
awscurl https://your-neptune-endpoint:port/propertygraph/statistics \
  --region us-east-1 \
  --service neptune-db
```

> **Note**
>
> Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -G https://your-neptune-endpoint:port/propertygraph/statistics
```

> **Note**
>
> Cet exemple n'inclut pas l'authentification IAM. Pour utiliser l'authentification IAM aveccurl, vous devez signer manuellement la demande. Envisagez d'utiliser AWS CLI le SDK ou à la awscurl place.

Les exemples suivants permettent de lancer manuellement la génération de statistiques par authentification :

AWS CLI

```
aws neptunedata manage-propertygraph-statistics \
```

```
  --endpoint-url https://your-neptune-endpoint:port \
  --mode refresh
```

SDK

```
import boto3
import json
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.manage_propertygraph_statistics(mode='refresh')

print(json.dumps(response['payload'], indent=2, default=str))
```

awscurl

Créez un fichier JSON nommé `request.json` qui contient :

```
{ "mode" : "refresh" }
```

Exécutez ensuite :

```
awscurl https://your-neptune-endpoint:port/propertygraph/statistics \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d @request.json
```

curl

```
curl -X POST https://your-neptune-endpoint:port/propertygraph/statistics \
  -H 'Content-Type: application/json' \
  -d '{ "mode" : "refresh" }'
```

> **Note**
>
> Cet exemple n'inclut pas l'authentification IAM. Pour utiliser l'authentification IAM aveccurl, vous devez signer manuellement la demande. Envisagez d'utiliser AWS CLI le SDK ou à la awscurl place.

Suppression des statistiques DFE

Vous pouvez supprimer toutes les statistiques de la base de données comme suit :

> **Note**
>
> Les exemples suivants utilisent le point de terminaison et les commandes du graphe de propriétés. Pour les données RDF, utilisez-les `delete-sparql-statistics` pour les points de terminaison REST et `/rdf/statistics` pour les points de terminaison. AWS CLI

AWS CLI

```
aws neptunedata delete-propertygraph-statistics \
  --endpoint-url https://your-neptune-endpoint:port
```

Pour plus d'informations, consultez delete-propertygraph-statistics dans le manuel de référence des commandes. AWS CLI

SDK

```
import boto3
import json
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.delete_propertygraph_statistics()
```

```
print(json.dumps(response['payload'], indent=2, default=str))
```

awscurl

```
awscurl https://your-neptune-endpoint:port/propertygraph/statistics \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultez[Utiliser awscurl avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée](#).

curl

```
curl -X DELETE https://your-neptune-endpoint:port/propertygraph/statistics
```

Voici les codes de retour HTTP valides :

- `200` : la suppression a été effectuée avec succès.

  Dans ce cas, une réponse typique ressemblerait à ce qui suit :

```
{
  "status" : "200 OK",
  "payload" : {
      "active" : false,
      "statisticsId" : -1
  }
}
```

- `204` : il n'y avait aucune statistique à supprimer.

  Dans ce cas, la réponse est vide (aucune réponse).

Si vous envoyez une demande de suppression à un point de terminaison de statistiques sur un nœud de lecteur, une exception `ReadOnlyViolationException` est renvoyée.

Codes d'erreur courants pour les demandes de statistiques DFE

Voici une liste des erreurs courantes qui peuvent survenir lorsque vous envoyez une demande à un point de terminaison de statistiques :

- `AccessDeniedException` – Code de retour : `400`. Message : `Missing Authentication Token`.
- `BadRequestException` (pour Gremlin et openCypher) : code de retour : `400`. Message : `Bad route: /pg/statistics`.
- `BadRequestException` (pour les données RDF) : code de retour : `400`. Message : `Bad route: /rdf/statistics`.
- `InvalidParameterException` – Code de retour : `400`. Message : `Statistics command parameter 'mode' has unsupported value '`*`the invalid value`*`'`.
- `MissingParameterException` – Code de retour : `400`. Message : `Content-type header not specified.`.
- `ReadOnlyViolationException` – Code de retour : `400`. Message : `Writes are not permitted on a read replica instance`.

Par exemple, si vous effectuez une demande alors que le DFE et les statistiques ne sont pas activés, vous obtiendrez une réponse comme celle-ci :

```
{
  "code" : "BadRequestException",
  "requestId" : "b2b8f8ee-18f1-e164-49ea-836381a3e174",
  "detailedMessage" : "Bad route: /sparql/statistics"
}
```

Génération d'un rapport récapitulatif rapide de votre graphe

L'API de résumé de graphe récupère les informations suivantes concernant votre graphe :

- Pour les graphes de propriétés (PG), l'API de résumé de graphe renvoie une liste en lecture seule des étiquettes et des clés de propriété des nœuds et des arêtes, ainsi que le nombre de nœuds, d'arêtes et de propriétés.
- Pour les graphes RDF (Resource Description Framework), l'API de résumé de graphe renvoie une liste en lecture seule de classes et de clés de prédicat, ainsi que le nombre de quadruplets, de sujets et de prédicats.

> **Note**
>
> L'API de résumé de graphe a été introduite dans la version 1.2.1.0 du moteur Neptune.

Grâce à l'API de résumé de graphe, vous pouvez rapidement vous faire une idée de la taille et du contenu de vos données de graphe. Vous pouvez également utiliser l'API de manière interactive dans un bloc-notes Neptune grâce à la magie de workbench Neptune `%summary`. Dans une application de graphe, l'API peut être utilisée pour améliorer les résultats de recherche en fournissant les étiquettes de nœuds ou d'arêtes découvertes dans le cadre de la recherche.

Les données récapitulatives de graphe sont tirées des statistiques DFE calculées par le moteur DFE Neptune pendant l'exécution et sont disponibles chaque fois que des statistiques DFE sont disponibles. Les statistiques sont activées par défaut lorsque vous créez un cluster de bases de données Neptune.

> **Note**
>
> La génération de statistiques est désactivée sur les types d'instance `t3` et `t4` (c'est-à-dire sur les types d'instance `db.t3.medium` et `db.t4g.medium`) pour économiser de la mémoire. Par conséquent, les données récapitulatives de graphe ne sont pas disponibles non plus sur ces types d'instances.

Vous pouvez vérifier le statut des statistiques DFE à l'aide de l'API de statut des statistiques. Tant que la génération automatique des statistiques n'est pas désactivée, les statistiques sont automatiquement mises à jour périodiquement.

Si vous voulez vous assurer que les statistiques sont aussi à jour que possible lorsque vous demandez un résumé de graphe, vous pouvez déclencher manuellement une mise à jour des statistiques juste avant de récupérer ce résumé. Si le graphe change pendant le calcul des statistiques, celles-ci prendront nécessairement un peu de temps, mais pas beaucoup.

Utilisation de l'API de résumé de graphe pour récupérer les informations récapitulatives d'un graphe

Pour un graphe de propriétés que vous interrogez à l'aide de Gremlin ou d'openCypher, vous pouvez récupérer un résumé de graphe à partir du point de terminaison du graphe de propriétés. Il existe à la fois un URI long et un URI court pour ce point de terminaison :

- `https://`*`your-neptune-host`*`:`*`port`*`/propertygraph/statistics/summary`
- `https://`*`your-neptune-host`*`:`*`port`*`/pg/statistics/summary`

Pour un graphe RDF que vous interrogez à l'aide de SPARQL, vous pouvez récupérer un résumé du graphe depuis le point de terminaison du résumé RDF :

- `https://`*`your-neptune-host`*`:`*`port`*`/rdf/statistics/summary`

Ces points de terminaison sont en lecture seule et ne prennent en charge qu'une opération HTTP `GET`. Si $GRAPH_SUMMARY_ENDPOINT est défini sur l'adresse du point de terminaison que vous souhaitez interroger, vous pouvez récupérer les données récapitulatives à l'aide de `curl` et du protocole HTTP `GET` comme suit :

```
curl -G "$GRAPH_SUMMARY_ENDPOINT"
```

Si aucune statistique n'est disponible lorsque vous essayez de récupérer un résumé de graphe, la réponse ressemble à ce qui suit :

```
{
  "detailedMessage": "Statistics are not available. Summary can only be generated after
 statistics are available.",
  "requestId": "48c1f788-f80b-b69c-d728-3f6df579a5f6",
```

```
  "code": "StatisticsNotAvailableException"
}
```

Le paramètre de requête d'URL du **`mode`** pour l'API de résumé du graphe

L'API de résumé de graphe accepte un paramètre de requête URL nommémode, qui peut avoir l'une des deux valeurs suivantes, à savoir `basic` (par défaut) et `detailed`. Pour un graphe RDF, la réponse de résumé de graphe en mode `detailed` contient un champ `subjectStructures` supplémentaire. Pour un graphe de propriétés, la réponse détaillée de résumé de graphe contient deux champs supplémentaires, à savoir `nodeStructures` et `edgeStructures`.

Pour demander une réponse de résumé de graphe sous forme `detailed`, incluez le paramètre `mode` comme suit :

```
curl -G "$GRAPH_SUMMARY_ENDPOINT?mode=detailed"
```

Si le paramètre `mode` n'est pas présent, le mode `basic` est utilisé par défaut. Il est donc possible de spécifier `?mode=basic` explicitement, mais cela n'est pas nécessaire.

Réponse de résumé pour un graphe de propriétés (PG)

Pour un graphe de propriétés vide, la réponse détaillée de résumé de graphe se présente comme suit :

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-01-10T07:58:47.972Z",
    "graphSummary" : {
      "numNodes" : 0,
      "numEdges" : 0,
      "numNodeLabels" : 0,
      "numEdgeLabels" : 0,
      "nodeLabels" : [ ],
      "edgeLabels" : [ ],
      "numNodeProperties" : 0,
      "numEdgeProperties" : 0,
      "nodeProperties" : [ ],
      "edgeProperties" : [ ],
```

```
      "totalNodePropertyValues" : 0,
      "totalEdgePropertyValues" : 0,
      "nodeStructures" : [ ],
      "edgeStructures" : [ ]
    }
  }
}
```

Une réponse de résumé de graphe de propriétés (PG) contient les champs suivants :

- **`status`** : code de retour HTTP de la demande. Si la demande aboutit, le code est 200.

  Pour obtenir une liste des erreurs courantes, consultez Erreurs courantes liées au résumé de graphe.
- **`payload`**
  - **`version`** : version de la réponse de résumé de graphe.
  - **`lastStatisticsComputationTime`** : horodatage, au format ISO 8601, indiquant l'heure à laquelle Neptune a calculé ses statistiques pour la dernière fois.
  - **`graphSummary`**
    - **`numNodes`** : nombre de nœuds dans le graphe.
    - **`numEdges`** : nombre d'arêtes dans le graphe.
    - **`numNodeLabels`** : nombre d'étiquettes de nœuds distinctes dans le graphe.
    - **`numEdgeLabels`** : nombre d'étiquettes d'arête distinctes dans le graphe.
    - **`nodeLabels`** : liste des étiquettes de nœuds distincts dans le graphe.
    - **`edgeLabels`** : liste des étiquettes d'arête distinctes dans le graphe.
    - **`numNodeProperties`** : nombre de propriétés de nœuds distinctes dans le graphe.
    - **`numEdgeProperties`** : nombre de propriétés d'arêtes distinctes dans le graphe.
    - **`nodeProperties`** : liste des propriétés de nœud distinctes dans le graphe et nombre de nœuds où chaque propriété est utilisée.
    - **`edgeProperties`** : liste des propriétés d'arêtes distinctes dans le graphe et nombre d'arêtes où chaque propriété est utilisée.
    - **`totalNodePropertyValues`** : nombre total d'utilisations de toutes les propriétés de nœud.
    - **`totalEdgePropertyValues`** : nombre total d'utilisations de toutes les propriétés d'arête.
    - **`nodeStructures`** : ce champ n'est présent que lorsque *`mode=detailed`* est spécifié dans la demande. Contient une liste de structures de nœuds, chacune avec les champs suivants :

    - **`count`** : nombre de nœuds dotés de cette structure spécifique.
    - **`nodeProperties`** : liste des propriétés de nœud présentes dans cette structure spécifique.
    - **`distinctOutgoingEdgeLabels`** : liste des différentes étiquettes d'arête sortantes présentes dans cette structure spécifique.
  - **`edgeStructures`** : ce champ n'est présent que lorsque *`mode=detailed`* est spécifié dans la demande. Contient une liste de structures d'arêtes, chacune avec les champs suivants :
    - **`count`** : nombre d'arêtes dotées de cette structure spécifique.
    - **`edgeProperties`** : liste des propriétés d'arête présentes dans cette structure spécifique.

Réponse de résumé de graphe RDF

Pour un graphe RDF vide, la réponse détaillée de résumé de graphe se présente comme suit :

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-01-10T07:58:47.972Z",
    "graphSummary" : {
      "numDistinctSubjects" : 0,
      "numDistinctPredicates" : 0,
      "numQuads" : 0,
      "numClasses" : 0,
      "classes" : [ ],
      "predicates" : [ ],
      "subjectStructures" : [ ]
    }
  }
}
```

Une réponse de résumé de graphe RDF contient les champs suivants :

- **`status`** : code de retour HTTP de la demande. Si la demande aboutit, le code est 200.

  Pour obtenir une liste des erreurs courantes, consultez Erreurs courantes liées au résumé de graphe.
- **`payload`**
  - **`version`** : version de la réponse de résumé de graphe.

- **`lastStatisticsComputationTime`** : horodatage, au format ISO 8601, indiquant l'heure à laquelle Neptune a calculé ses statistiques pour la dernière fois.
- **`graphSummary`**
  - **`numDistinctSubjects`** : nombre de sujets distincts dans le graphe.
  - **`numDistinctPredicates`** : nombre de prédicats distincts dans le graphe.
  - **`numQuads`** : nombre de quadruplets dans le graphe.
  - **`numClasses`** : nombre de classes dans le graphe.
  - **`classes`** : liste des classes dans le graphe.
  - **`predicates`** : liste des prédicats dans le graphe et nombre de prédicats.
  - **`subjectStructures`** : ce champ n'est présent que lorsque *`mode=detailed`* est spécifié dans la demande. Contient une liste de structures de sujets, chacune avec les champs suivants :
    - **`count`** : nombre d'occurrences de cette structure spécifique.
    - **`predicates`** : liste des prédicats présents dans cette structure spécifique.

Exemple de réponse de résumé de graphe de propriétés (PG)

Voici la réponse détaillée de résumé de graphe de propriétés contenant un exemple de jeu de données de routes aériennes impliquant un graphe de propriétés :

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-03-01T14:35:03.804Z",
    "graphSummary" : {
      "numNodes" : 3748,
      "numEdges" : 51300,
      "numNodeLabels" : 4,
      "numEdgeLabels" : 2,
      "nodeLabels" : [
        "continent",
        "country",
        "version",
        "airport"
      ],
      "edgeLabels" : [
```

```
        "contains",
        "route"
      ],
      "numNodeProperties" : 14,
      "numEdgeProperties" : 1,
      "nodeProperties" : [
        {
          "desc" : 3748
        },
        {
          "code" : 3748
        },
        {
          "type" : 3748
        },
        {
          "country" : 3503
        },
        {
          "longest" : 3503
        },
        {
          "city" : 3503
        },
        {
          "lon" : 3503
        },
        {
          "elev" : 3503
        },
        {
          "icao" : 3503
        },
        {
          "region" : 3503
        },
        {
          "runways" : 3503
        },
        {
          "lat" : 3503
        },
        {
          "date" : 1
```

```
        },
        {
          "author" : 1
        }
      ],
      "edgeProperties" : [
        {
          "dist" : 50532
        }
      ],
      "totalNodePropertyValues" : 42773,
      "totalEdgePropertyValues" : 50532,
      "nodeStructures" : [
        {
          "count" : 3471,
          "nodeProperties" : [
            "city",
            "code",
            "country",
            "desc",
            "elev",
            "icao",
            "lat",
            "lon",
            "longest",
            "region",
            "runways",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [
            "route"
          ]
        },
        {
          "count" : 161,
          "nodeProperties" : [
            "code",
            "desc",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [
            "contains"
          ]
        },
```

```
        {
          "count" : 83,
          "nodeProperties" : [
            "code",
            "desc",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [ ]
        },
        {
          "count" : 32,
          "nodeProperties" : [
            "city",
            "code",
            "country",
            "desc",
            "elev",
            "icao",
            "lat",
            "lon",
            "longest",
            "region",
            "runways",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [ ]
        },
        {
          "count" : 1,
          "nodeProperties" : [
            "author",
            "code",
            "date",
            "desc",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [ ]
        }
      ],
      "edgeStructures" : [
        {
          "count" : 50532,
          "edgeProperties" : [
            "dist"
```

```
          ]
        }
      ]
    }
  }
}
```

Exemple de réponse de résumé de graphe RDF

Voici la réponse détaillée de résumé de graphe RDF contenant un [exemple de jeu de données de routes aériennes impliquant un graphe RDF] :

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-03-01T14:54:13.903Z",
    "graphSummary" : {
      "numDistinctSubjects" : 54403,
      "numDistinctPredicates" : 19,
      "numQuads" : 158571,
      "numClasses" : 4,
      "classes" : [
        "http://kelvinlawrence.net/air-routes/class/Version",
        "http://kelvinlawrence.net/air-routes/class/Airport",
        "http://kelvinlawrence.net/air-routes/class/Continent",
        "http://kelvinlawrence.net/air-routes/class/Country"
      ],
      "predicates" : [
        {
          "http://kelvinlawrence.net/air-routes/objectProperty/route" : 50656
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/dist" : 50656
        },
        {
          "http://kelvinlawrence.net/air-routes/objectProperty/contains" : 7004
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/code" : 3747
        },
        {
          "http://www.w3.org/2000/01/rdf-schema#label" : 3747
```

```
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/type" : 3747
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/desc" : 3747
        },
        {
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" : 3747
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/icao" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/lat" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/region" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/runways" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/longest" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/elev" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/lon" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/country" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/city" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/author" : 1
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/date" : 1
        }
      ],
```

```
    "subjectStructures" : [
      {
        "count" : 50656,
        "predicates" : [
          "http://kelvinlawrence.net/air-routes/datatypeProperty/dist"
        ]
      },
      {
        "count" : 3471,
        "predicates" : [
          "http://kelvinlawrence.net/air-routes/datatypeProperty/city",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/country",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/elev",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/icao",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/lat",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/lon",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/longest",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/region",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/runways",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
          "http://kelvinlawrence.net/air-routes/objectProperty/route",
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
          "http://www.w3.org/2000/01/rdf-schema#label"
        ]
      },
      {
        "count" : 238,
        "predicates" : [
          "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
          "http://kelvinlawrence.net/air-routes/objectProperty/contains",
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
          "http://www.w3.org/2000/01/rdf-schema#label"
        ]
      },
      {
        "count" : 31,
        "predicates" : [
          "http://kelvinlawrence.net/air-routes/datatypeProperty/city",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/country",
```

```
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/elev",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/icao",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lat",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lon",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/longest",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/region",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/runways",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 6,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 1,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/author",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/date",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        }
      ]
    }
  }
}
```

Utilisation Gestion des identités et des accès AWS Authentification (IAM) avec points de terminaison de synthèse graphique

Vous pouvez accéder aux points de terminaison de résumé de graphe en toute sécurité grâce à l'authentification IAM à l'aide d'awscurl ou de tout autre outil compatible avec HTTPS et IAM. Consultez Utiliser `awscurl` avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée pour découvrir comment configurer les informations d'identification appropriées. Une fois que y avez accès, vous pouvez effectuer des demandes comme celles-ci :

```
awscurl "$GRAPH_SUMMARY_ENDPOINT" \
    --region (your region) \
    --service neptune-db
```

> **Important**
>
> L'identité ou le rôle IAM qui crée les informations d'identification temporaires doit être associé à une politique IAM autorisant l'action GetGraphSummaryIAM.

Consultez Erreurs d'authentification IAM pour obtenir la liste des erreurs IAM courantes que vous pouvez rencontrer.

Codes d'erreur courants pouvant être renvoyés par une demande de résumé de graphe

Code d'erreur du service Neptune	Statu HTT	Message	Scénario d'erreur	Résolution
`AccessDeniedException`	403	Jeton d'authentification manquant.	Une demande non signée ou mal signée a été envoyée à la base de données Neptune où IAM est activé.	Signez la demande avec SigV4 avant de l'envoyer (voir IAM et résumés de graphe).

Code d'erreur du service Neptune	Statu HTT	Message	Scénario d'erreur	Résolution
	403	L'utilisateur : n'*(user ARN)* est pas autorisé à exécuter : neptune-db : on resource :GetGraph Summary . *(resource ARN)*	La politique IAM n'autorise pas cette action GetGraphS ummarylorsque la demande de résumé du graphe a été envoyée à la base de données Neptune avec IAM activé.	Assurez-vous que la politique IAM attachée à l'utilisa teur ou au rôle à l'origine de la demande autorise l'action `GetGraphS ummary` .
`BadReques tException`	400	Les statistiques étant désactivées, le résumé de graphe est également désactivé.	Essayer d'obtenir un résumé sur les types d'instances extensibl es (`t3` ou `t4g`) dont les statistiques sont désactivées.	Utilisez un type d'instance dans lequel la générati on de statistiques est activée (toutes les instances prises en charge sauf `t3` et `t4g`).
	400	Mauvais itinérair e : */rdf/stat istics/su mmarypathapi*	Demande envoyée à un chemin non valide.	Utilisez la route correcte pour le point de terminais on du résumé de graphe.
`InvalidPa rameterEx ception`	400	La demande contient des paramètre s inconnus : « *(unknown parameter or parameters)* ».	Survient lorsqu'un paramètre non valide est spécifié dans la demande.	N'utilisez que des paramètres valides (tels que `mode`) dans la demande.

Code d'erreur du service Neptune	Statu HTT	Message	Scénario d'erreur	Résolution
InvalidParameterException	400	La valeur « » du paramètre de requête URI « mode » n'est pas prise en charge*(invalid value)*.	Se produit lorsque le paramètre URL « mode » de la demande est suivi d'une valeur non valide.	Utilisez des valeurs valides (telles que `basic` ou `detailed`) lorsque vous spécifiez le paramètre URL « mode ».
MethodNotAllowedException	405	Méthode non autorisée	Appel du point de terminaison de résumé avec une méthode HTTP autre que `GET` (telle que `POST` ou `DELETE`).	Utilisez la méthode HTTP `GET` lorsque vous appelez le point de terminaison de résumé.
StatisticsNotAvailableException	400	Les statistiques ne sont pas encore calculées: Le résumé du graphe sera disponible une fois le calcul des statistiques terminé.	Aucune statistiq ue n'est disponible lorsque la demande est envoyée au point de terminaison de résumé.	Attendez que la génération des statistiques soit terminée. Vous pouvez vérifier le statut de la génération des statistiques à l'aide de l'API de statut des statistiques.
	400	La limite de statistiq ues est atteinte. Le résumé du graphe n'est donc pas disponible.	La génération de statistiques s'est arrêtée, car les limites de taille des statistiques ont été atteintes.	Le résumé n'est pas disponible sur ce graphe.

Par exemple, si vous soumettez une demande au point de terminaison de résumé de graphe dans une base de données Neptune sur laquelle l'authentification IAM est activée et que les autorisations nécessaires ne se trouvent pas dans la politique IAM du demandeur, vous obtiendrez une réponse comme celle-ci :

```
{
  "detailedMessage": "User: arn:aws:iam::(account ID):(user or user name) is not
 authorized to perform: neptune-db:GetGraphSummary on resource: arn:aws:neptune-
db:(region):(account ID):(cluster resource ID)/*",
  "requestId": "7ac2b98e-b626-d239-1d05-74b4c88fce82",
  "code": "AccessDeniedException"
}
```

Connectivité JDBC Amazon Neptune

Amazon Neptune a publié un pilote JDBC open source qui prend en charge les requêtes OpenCypher, G705 et SPARQL. SQL-Gremlin La connectivité JDBC facilite la connexion à Neptune grâce à des outils de business intelligence (BI) tels que Tableau. L'utilisation du pilote JDBC avec Neptune n'entraîne aucun coût supplémentaire : vous ne payez que les ressources Neptune consommées.

Ce pilote est compatible avec JDBC 4.2 et nécessite au moins Java 8. Pour en savoir plus sur l'utilisation d'un pilote JDBC, consultez la documentation relative à l'API JDBC.

Le GitHub projet, dans lequel vous pouvez signaler des problèmes et ouvrir des demandes de fonctionnalités, contient une documentation détaillée sur le pilote :

Pilote JDBC pour Amazon Neptune

- Utilisation de SQL avec le pilote JDBC
- Utilisation de Gremlin avec le pilote JDBC
- Utilisation d'openCypher avec le pilote JDBC
- Utilisation de SPARQL avec le pilote JDBC

Premiers pas avec le pilote Neptune JDBC

Pour utiliser le pilote JDBC Neptune afin de vous connecter à une instance Neptune, soit le pilote JDBC doit être déployé sur une instance Amazon EC2 située dans le même VPC que le cluster de bases de données Neptune, soit l'instance doit être disponible via un tunnel SSH ou un équilibreur de charge. Un tunnel SSH peut être configuré dans le pilote en interne ou en externe.

Vous pouvez télécharger le pilote source ici. Le pilote est fourni sous la forme d'un seul fichier JAR avec un nom tel que `neptune-jdbc-1.0.0-all.jar`. Pour l'utiliser, placez le fichier JAR dans le dossier `classpath` de votre application. Ou, si votre application utilise Maven ou Gradle, vous pouvez utiliser les commandes Maven ou Gradle appropriées pour installer le pilote à partir du fichier JAR.

Le pilote a besoin d'une URL de connexion JDBC pour se connecter à Neptune, sous la forme suivante :

```
jdbc:neptune:(connection
 type)://(host);property=value;property=value;...;property=value
```

Les sections relatives à chaque langage de requête du GitHub projet décrivent les propriétés que vous pouvez définir dans l'URL de connexion JDBC pour ce langage de requête.

Si le fichier JAR se trouve dans le dossier `classpath` de votre application, aucune autre configuration n'est nécessaire. Vous pouvez connecter le pilote à l'aide de l'interface JDBC `DriverManager` et d'une chaîne de connexion Neptune. Par exemple, si le cluster de bases de données Neptune est accessible via le point de terminaison `neptune-example.com` sur le port 8182, vous pouvez vous connecter à openCypher comme suit :

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

void example() {
    String url = "jdbc:neptune:opencypher://bolt://neptune-example:8182";

    Connection connection = DriverManager.getConnection(url);
    Statement statement = connection.createStatement();

    connection.close();
}
```

Les sections de documentation du GitHub projet pour chaque langage de requête décrivent comment construire la chaîne de connexion lors de l'utilisation de ce langage de requête.

Utilisation de Tableau avec le pilote Neptune JDBC

Pour utiliser Tableau avec le pilote Neptune JDBC, commencez par télécharger et installer la version la plus récente de Tableau Desktop. Téléchargez le fichier JAR du pilote Neptune JDBC, ainsi que le fichier du connecteur Neptune Tableau (fichier `.taco`).

Pour vous connecter à Tableau pour Neptune sur un Mac

1. Placez le fichier JAR du pilote JDBC Neptune dans le dossier `/Users/(your user name)/Library/Tableau/Drivers`.
2. Placez le fichier `.taco` du connecteur Neptune Tableau dans le dossier `/Users/(your user name)/Documents/My Tableau Repository/Connectors`.

3. Si l'authentification IAM est activée, configurez l'environnement en conséquence. Notez que les variables d'environnement définies dans `.zprofile/`, `.zshenv/`, `.bash_profile`, etc. ne fonctionnent pas. Les variables d'environnement doivent être définies de manière à pouvoir être chargées par une application graphique.

   Pour définir vos informations d'identification, vous pouvez placer votre clé d'accès et votre clé secrète dans le fichier `/Users/(your user name)/.aws/credentials`.

   Pour définir facilement la région de service, ouvrez un terminal et entrez la commande suivante, en utilisant la région de votre application (par exemple, `us-east-1`) :

   ```
   launchctl setenv SERVICE_REGION region name
   ```

   Il existe d'autres moyens de définir des variables d'environnement qui persistent après un redémarrage, mais quelle que soit la technique utilisée, vous devez définir des variables accessibles à une application graphique.

4. Pour charger des variables d'environnement dans une interface graphique sur Mac, entrez cette commande sur un terminal :

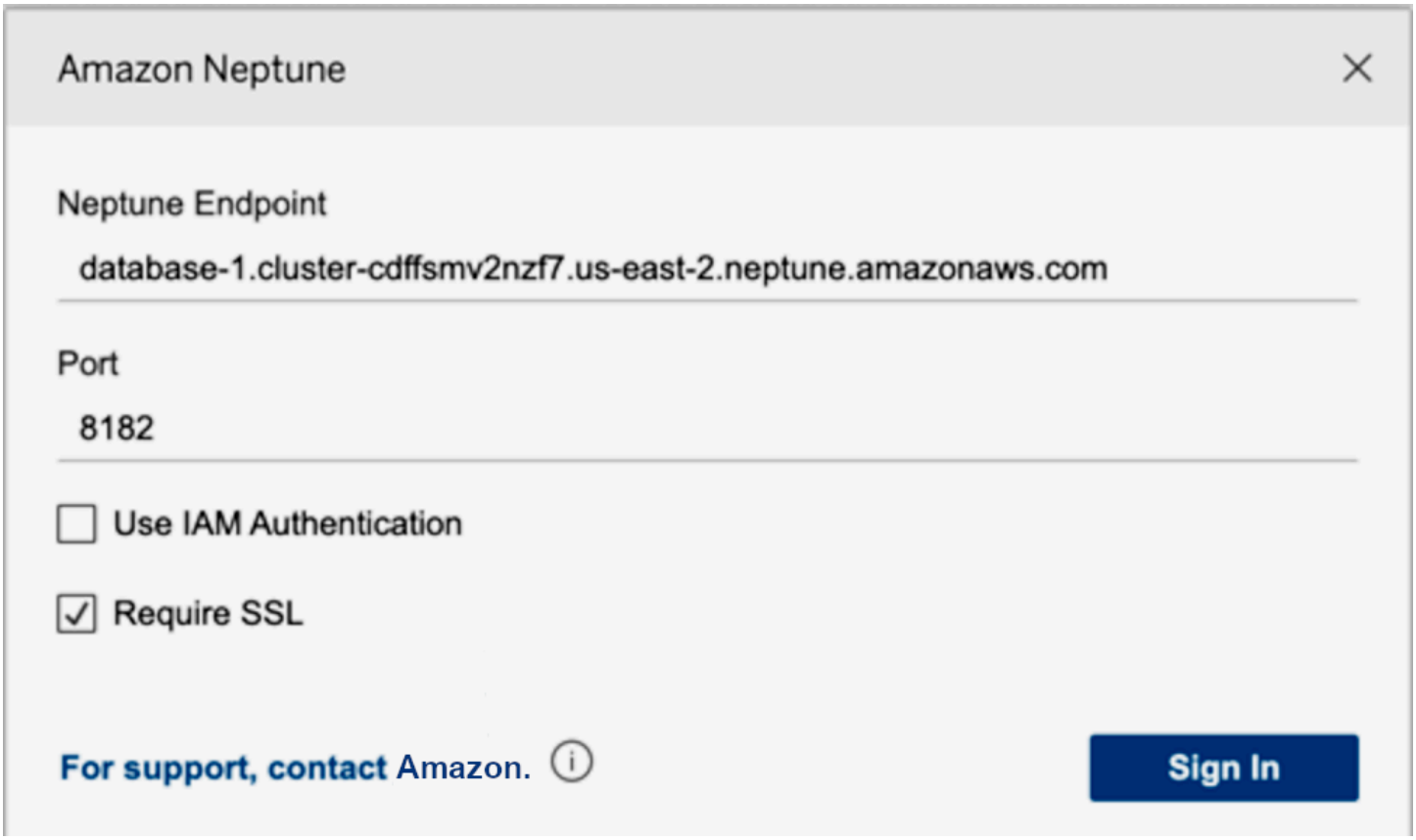
   ```
   /Applications/Tableau/Desktop/2021.1.app/Contents/MacOS/Tableau
   ```

Pour vous connecter à Tableau pour Neptune sur un ordinateur Windows

1. Placez le fichier JAR du pilote JDBC Neptune dans le dossier `C:\Program Files\Tableau \Drivers`.
2. Placez le fichier `.taco` du connecteur Neptune Tableau dans le dossier `C:\Users\(your user name)\Documents\My Tableau Repository\Connectors`.
3. Si l'authentification IAM est activée, configurez l'environnement en conséquence.

   Cela peut être aussi simple que de définir des variables d'environnement utilisateur `ACCESS_KEY`, `SECRET_KEY` et `SERVICE_REGION`.

Une fois Tableau ouvert, sélectionnez Plus du côté gauche de la fenêtre. Si le fichier du connecteur Tableau est correctement localisé, vous pouvez sélectionner Amazon Neptune d' AWS dans la liste qui apparaît :



Vous ne devriez pas avoir à modifier le port ni à ajouter d'options de connexion. Entrez le point de terminaison Neptune et définissez votre configuration IAM et SSL (vous devez activer le protocole SSL si vous utilisez IAM).

Lorsque vous sélectionnez Se connecter, la connexion peut prendre plus de 30 secondes si le graphe est volumineux. Tableau collecte les tables de sommets et d'arêtes, joint les sommets sur des arêtes et crée des visualisations.

Résolution des problèmes de connexion à un pilote JDBC

Si le pilote ne parvient pas à se connecter au serveur, utilisez la fonction `isValid` de l'objet JDBC `Connection` pour vérifier si la connexion est valide. Si la fonction renvoie `false`, ce qui signifie que la connexion n'est pas valide, vérifiez que le point de terminaison auquel vous êtes connecté est correct et que vous êtes dans le VPC du cluster de bases de données Neptune ou que vous disposez d'un tunnel SSH valide vers le cluster.

Si vous recevez une réponse `No suitable driver found for` *`(connection string)`* à l'appel `DriverManager.getConnection`, il existe probablement un problème au début de la chaîne de connexion. Assurez-vous que la chaîne de connexion commence comme suit :

```
jdbc:neptune:opencypher://...
```

Pour recueillir plus d'informations sur la connexion, vous pouvez ajouter un `LogLevel` à la chaîne de connexion comme suit :

```
jdbc:neptune:opencypher://(JDBC URL):(port);logLevel=trace
```

Vous pouvez également ajouter `properties.put("logLevel", "trace")` dans les propriétés d'entrée pour enregistrer les informations de suivi.

Mises à jour du moteur Amazon Neptune

Amazon Neptune publie régulièrement des mises à jour du moteur. Pour déterminer la version du moteur qui est actuellement installée, vous pouvez utiliser l'API instance-status.

Les versions du moteur sont répertoriées dans Versions du moteur pour Amazon Neptune et les correctifs apparaissent dans Dernières mises à jour.

Vous trouverez plus d'informations sur le mode de publication des mises à jour et sur la mise à niveau du moteur Neptune dans votre base de données sous Maintenance de cluster. Par exemple, la numérotation des versions du moteur est expliquée dans Numéros de version du moteur.

Gestion des exceptions et nouvelles tentatives

Pour créer des applications robustes sur Neptune, il faut souvent se préparer aux imprévus, notamment lorsqu'il s'agit de gérer les erreurs renvoyées par la base de données. L'une des réponses les plus courantes aux exceptions côté serveur consiste à réessayer l'opération qui a échoué. Bien que la logique des nouvelles tentatives soit essentielle à la résilience des systèmes, vous devez reconnaître que toutes les erreurs ne doivent pas être traitées de la même manière. Plutôt que de vous fier à des comportements de nouvelle tentative génériques, une approche réfléchie peut vous aider à créer des applications plus fiables et plus efficaces.

Pourquoi la logique des nouvelles tentatives est importante

La logique de nouvelle tentative est un élément essentiel de toute application distribuée. Des problèmes transitoires tels que l'instabilité du réseau, des contraintes de ressources temporaires ou des conflits de modification simultanés peuvent entraîner l'échec des opérations. Dans de nombreux cas, ces échecs n'indiquent pas un problème permanent et peuvent être résolus en attendant et en réessayant. La mise en œuvre d'une stratégie de relance solide tient compte de la réalité des environnements imparfaits dans les systèmes distribués, garantissant ainsi une fiabilité et une continuité accrues avec moins d'interventions manuelles.

Les risques de nouvelles tentatives indiscriminées

Réessayer chaque erreur par défaut peut avoir plusieurs conséquences imprévues :

- Contentions accrues : lorsque des opérations qui échouent en raison d'une forte simultanéité sont réessayées à plusieurs reprises, le contentieux général peut s'aggraver. Cela peut entraîner un cycle d'échecs de transactions et de dégradation des performances.
- Épuisement des ressources — Les tentatives indiscriminées peuvent consommer des ressources système supplémentaires, tant du côté du client que du côté du serveur. Cela peut potentiellement entraîner un ralentissement ou même une dégradation du service.
- Latence accrue pour les clients — Des tentatives excessives peuvent entraîner des retards importants pour les applications clientes, en particulier si chaque nouvelle tentative implique des périodes d'attente. Cela peut avoir un impact négatif sur l'expérience utilisateur et les processus en aval.

Élaboration d'une stratégie de réessai pratique

Pour créer une application résiliente et efficace, développez une stratégie de nouvelle tentative adaptée aux conditions d'erreur spécifiques que votre application peut rencontrer. Voici quelques éléments à prendre en compte pour orienter votre approche :

- Identifiez les erreurs réessayables : toutes les exceptions ne doivent pas être réessayées. Par exemple, les erreurs de syntaxe, les échecs d'authentification ou les requêtes non valides ne doivent pas déclencher une nouvelle tentative. Neptune fournit des codes d'erreur et des recommandations générales pour lesquels vous pouvez réessayer en toute sécurité, mais vous devez implémenter la logique adaptée à votre cas d'utilisation.
- Implémenter un délai d'attente exponentiel : pour les erreurs transitoires, utilisez une stratégie de temporisation exponentielle afin d'augmenter progressivement le temps d'attente entre les tentatives. Cela permet d'atténuer les conflits et de réduire le risque de défaillances en cascade.
- Tenez compte de la durée de la pause initiale : une première tentative trop rapide peut se terminer par la même erreur si le serveur n'a pas eu le temps de libérer les ressources dont la requête a besoin pour réussir. Une pause plus longue dans les bonnes situations pourrait réduire le gaspillage de demandes et la pression du serveur.
- Ajoutez de la nervosité au temps perdu : bien que le ralentissement exponentiel soit efficace, il peut tout de même entraîner des tempêtes de tentatives synchronisées si de nombreux clients échouent en même temps puis réessaient ensemble. L'ajout de gigter, une petite variation aléatoire du délai d'attente, permet d'étaler les tentatives de relance, réduisant ainsi le risque que tous les clients réessayent simultanément et provoquent un nouveau pic de charge.
- Limiter les tentatives : définissez un nombre maximum raisonnable de tentatives pour éviter les boucles infinies et l'épuisement des ressources.
- Surveiller et ajuster — Surveillez en permanence le taux d'erreur de votre application et ajustez votre stratégie de réessai en fonction des besoins. Si vous remarquez un nombre élevé de tentatives pour une opération donnée, déterminez si l'opération peut être optimisée ou sérialisée.

Pour une analyse plus approfondie du décalage exponentiel et de l'instabilité en tant que modèle général de conception du cloud, voir Réessayer avec un ralentissement dans les directives prescriptives. AWS

Exemples de scénarios

La bonne stratégie de relance dépend de la nature de l'échec, de la charge de travail et des modèles d'erreur que vous observez. Le tableau suivant résume certains scénarios d'échec courants et explique comment les considérations relatives à la stratégie de nouvelle tentative s'appliquent à chacun d'entre eux. Les paragraphes explicatifs suivants fournissent un contexte supplémentaire.

Scénario	Récupérable ?	Retard et instabilité	Pause initiale	Limite de tentatives	Surveiller et régler
CME occasionnel sur des requêtes courtes	Oui	Retournez rapidement, ajoutez de la nervosité	Courte (par exemple, 100 ms)	Élevée	Surveillez la hausse des taux CME
CME fréquent sur les requêtes Longer-Running	Oui	Temps d'arrêt plus long, augmentation de la nervosité	Plus long (par exemple, 2 secondes)	Modérée	Étudier et réduire les contentieux
Limites de mémoire pour les requêtes coûteuses	Oui	Longue période de recul	Longue (par exemple, 5 à 10 secondes)	Faible	Optimisation de la requête, alerte en cas de persistance
Délai d'attente pour les requêtes modérées	Peut-être	Atténuation modérée, augmentation de la nervosité	Modéré (par exemple, 1 s)	Faible à modérée	Évaluer la charge du serveur et la conception des requêtes

Scénario 1 : CME occasionnel sur de courtes requêtes

Pour une charge de travail qui `ConcurrentModificationException` apparaît rarement lors de mises à jour courtes et simples, ces erreurs sont généralement transitoires et peuvent être réessayées en toute sécurité. Faites une courte pause initiale (par exemple, 100 millisecondes) avant la première tentative. Cette durée permet à tout bref verrou de s'effacer. Combinez cela à un court recul exponentiel et à une instabilité pour éviter les nouvelles tentatives synchronisées. Le coût d'une nouvelle tentative étant faible, une limite de nouvelles tentatives plus élevée est raisonnable. Néanmoins, surveillez le taux de CME pour détecter toute tendance à une augmentation de la contention dans vos données.

Scénario 2 : CME fréquent sur des requêtes de longue durée

Si votre application reçoit fréquemment des CME sur des requêtes de longue durée, cela suggère une contention plus grave. Dans ce cas, commencez par une pause initiale plus longue (par exemple, 2 secondes), pour que la requête en cours contenant le verrou ait suffisamment de temps pour se terminer. Utilisez un recul exponentiel plus long et ajoutez de l'instabilité. Limitez le nombre de tentatives afin d'éviter les retards excessifs et l'utilisation excessive des ressources. Si le conflit persiste, examinez votre charge de travail pour détecter des modèles et envisagez de sérialiser les mises à jour ou de réduire la simultanéité pour en traiter la cause première.

Scénario 3 : limites de mémoire pour les requêtes coûteuses

Lorsque des erreurs liées à la mémoire se produisent au cours d'une requête connue gourmande en ressources, il peut être judicieux de réessayer, mais uniquement après une longue pause initiale (par exemple, 5 à 10 secondes ou plus) pour permettre au serveur de libérer des ressources. Utilisez une stratégie d'attente prolongée et définissez une faible limite de tentatives, car il est peu probable que les échecs répétés soient résolus sans modification de la requête ou de la charge de travail. Les erreurs persistantes devraient déclencher des alertes et entraîner un examen de la complexité des requêtes et de l'utilisation des ressources.

Scénario 4 : délai d'expiration pour les requêtes modérées

Un délai d'attente pour une requête modérément coûteuse est un cas plus ambigu. Parfois, une nouvelle tentative peut réussir si le délai d'attente est dû à un pic temporaire de charge du serveur ou aux conditions du réseau. Commencez par une pause initiale modérée (1 seconde, par exemple) pour permettre au système de récupérer. Appliquez un ralentissement modéré et ajoutez de la gigue pour éviter les nouvelles tentatives synchronisées. Maintenez la limite de tentatives faible à modérée, car des délais d'attente répétés peuvent indiquer un problème plus grave lié à la requête ou à la

capacité du serveur. Surveillez les modèles : si les délais deviennent fréquents, déterminez si la requête doit être optimisée ou si le cluster Neptune est sous-approvisionné.

Surveillance et observabilité

La surveillance est un élément essentiel de toute stratégie de nouvelle tentative. Une observabilité efficace vous aide à comprendre dans quelle mesure votre logique de nouvelle tentative fonctionne et fournit des signaux précoces lorsqu'un élément de votre charge de travail ou de la configuration de votre cluster nécessite une attention particulière.

MainRequestQueuePendingRequests

Cette CloudWatch métrique permet de suivre le nombre de demandes en attente dans la file d'entrée de Neptune. Une valeur croissante indique que les requêtes sont sauvegardées, ce qui peut être le signe d'un contentieux excessif, d'un sous-provisionnement des ressources ou d'une tempête de nouvelles tentatives. La surveillance de cette métrique vous aide à identifier les situations dans lesquelles votre stratégie de nouvelle tentative est à l'origine ou aggrave les problèmes de mise en file d'attente, et peut vous inviter à ajuster votre approche avant que les échecs ne se multiplient.

Autres CloudWatch mesures

D'autres indicateurs Neptune`CPUUtilization`, tels que, et la latence des requêtes`TotalRequestsPerSecond`, fournissent un contexte supplémentaire. Par exemple, un processeur élevé I/O associé à des files d'attente de plus en plus longues peut indiquer que votre cluster est surchargé ou que les requêtes sont trop volumineuses ou trop fréquentes. CloudWatch des alarmes peuvent être définies sur ces métriques pour vous avertir d'un comportement anormal et vous aider à corréler les pics d'erreurs ou de nouvelles tentatives avec les contraintes de ressources sous-jacentes.

API Neptune Status et Query

L'API Neptune Status pour G705 et ses API analogues pour SPARQL fournissent une vue en temps réel des requêtes acceptées OpenCypheret exécutées sur le cluster, ce qui est utile pour diagnostiquer les goulots d'étranglement ou comprendre l'impact de la logique de nouvelle tentative en temps réel.

En combinant ces outils de surveillance, vous pouvez :

- Détectez les tentatives qui contribuent à la mise en file d'attente et à la dégradation des performances.

- Identifiez à quel moment il convient de dimensionner votre cluster Neptune ou d'optimiser les requêtes.
- Vérifiez que votre stratégie de nouvelle tentative résout les échecs transitoires sans masquer les problèmes plus graves.
- Recevez des alertes précoces en cas de conflits émergents ou d'épuisement des ressources.

La surveillance et les alertes proactives sont essentielles au bon déroulement du déploiement de Neptune, en particulier à mesure que la simultanéité et la complexité de votre application augmentent.

Sécurisation de votre base de données Amazon Neptune

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cette notion par les termes sécurité du cloud et sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Third-party les auditeurs testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de AWS conformité. Pour en savoir plus sur les programmes de conformité qui s'appliquent à Amazon Neptune, consultez Services AWS concernés par le programme de conformité.
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris de la sensibilité de vos données, des exigences de votre entreprise, ainsi que de la législation et de la réglementation applicables.

Cette documentation vous aidera à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation de Neptune. Les rubriques suivantes expliquent comment configurer Neptune pour répondre à vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui vous aident à surveiller et à sécuriser vos ressources Neptune.

Rubriques

- Mises à niveau du système d'exploitation Amazon Neptune
- Protection des données de votre base de données Amazon Neptune
- Authentification de votre base de données Amazon Neptune avec Gestion des identités et des accès AWS
- Activation de l'authentification de base de données IAM dans Amazon Neptune
- Connexion à votre base de données Amazon Neptune à l'aide de Gestion des identités et des accès AWS authentification
- Gestion de l'accès aux bases de données Amazon Neptune à l'aide des politiques IAM
- Utilisation de rôles liés à un service pour Amazon Neptune
- Utilisation d'informations d'identification temporaires pour se connecter à Amazon Neptune

- Enregistrement et surveillance de l'utilisation et des performances dans Amazon Neptune
- Amazon Neptune et points de terminaison VPC d'interface (AWS PrivateLink)
- Considérations relatives à la conformité pour Amazon Neptune
- Création de déploiements Amazon Neptune résilients et tolérants aux catastrophes

Mises à niveau du système d'exploitation Amazon Neptune

Amazon Neptune garantit l'amélioration continue des performances, de la sécurité et de la stabilité des bases de données grâce à des mises à niveau régulières du système d'exploitation. Ces mises à niveau sont appliquées à la fois aux bases de données Neptune et à Neptune Analytics, selon un processus de mise à jour structuré. Amazon Neptune publie des mises à niveau du système d'exploitation au moins une fois par mois.

Les mises à jour du système d'exploitation de base de données Neptune qui ne nécessitent pas de temps d'arrêt sont automatiquement appliquées pendant la période de maintenance. Certaines mises à jour du système d'exploitation (par exemple : mise à niveau de la version du noyau) nécessitent le redémarrage de l'instance. Ces mises à jour sont facultatives et ne sont pas soumises à une date précise pour les appliquer. Toutefois, si vous n'appliquez pas ces mises à jour, elles peuvent éventuellement devenir obligatoires et automatiquement appliquées pendant la période de maintenance de votre instance.

Neptune Analytics - Les mises à niveau du système d'exploitation de Neptune Analytics sont fluides et ne nécessitent aucune intervention du client. Ces mises à jour sont automatiquement appliquées en arrière-plan sans interruption.

Pour garantir la sécurité et la conformité, nous vous recommandons d'appliquer régulièrement toutes les mises à jour mises à disposition par Amazon Neptune pendant votre période de maintenance. Le fait de se tenir au courant de toutes les mises à jour facultatives et obligatoires permet d'intégrer les correctifs de sécurité essentiels et de garantir le respect des diverses obligations de conformité. Les versions obsolètes du système d'exploitation peuvent entraîner le non-respect des exigences réglementaires.

Minimiser les temps d'arrêt pour les mises à niveau du système d'exploitation nécessitant un redémarrage

Pour les mises à niveau du système d'exploitation qui nécessitent un redémarrage, nous vous recommandons de mettre à jour d'abord les instances du lecteur dans un cluster, puis l'instance du

rédacteur afin d'optimiser la disponibilité de votre cluster. Nous ne recommandons pas de mettre à jour les instances du lecteur et du rédacteur en même temps, car vous pourriez subir des temps d'arrêt plus longs en cas de basculement.

Appliquer des mises à niveau du système d'exploitation à votre instance de base de données Neptune

Les instances de base de données Neptune nécessitent parfois des mises à jour du système d'exploitation. Amazon Neptune met à niveau le système d'exploitation vers une version plus récente afin d'améliorer les performances de la base de données et le niveau de sécurité global des clients. En général, les mises à jour prennent environ 10 minutes. Les mises à jour du système d'exploitation ne modifient pas la version du moteur de base de données ou la classe d'instance de base de données d'une instance de base de données.

Pour être averti lorsqu'une nouvelle mise à jour facultative est disponible, vous pouvez vous abonner `RDS-EVENT-0230` à la catégorie des événements relatifs aux correctifs de sécurité. Pour plus d'informations sur l'abonnement aux événements Amazon Neptune, consultez la section Abonnement aux notifications d'événements Neptune.

> **Important**
>
> Votre instance de base de données Amazon Neptune sera mise hors ligne lors de la mise à niveau du système d'exploitation. Vous pouvez minimiser les temps d'arrêt du cluster en utilisant un cluster multi-instances. Si vous ne possédez pas de cluster multi-instances, vous pouvez choisir d'en créer un temporairement en ajoutant des instances secondaires pour effectuer cette maintenance, puis en supprimant les instances de lecteur supplémentaires une fois la maintenance terminée (les frais habituels pour l'instance secondaire s'appliqueront).

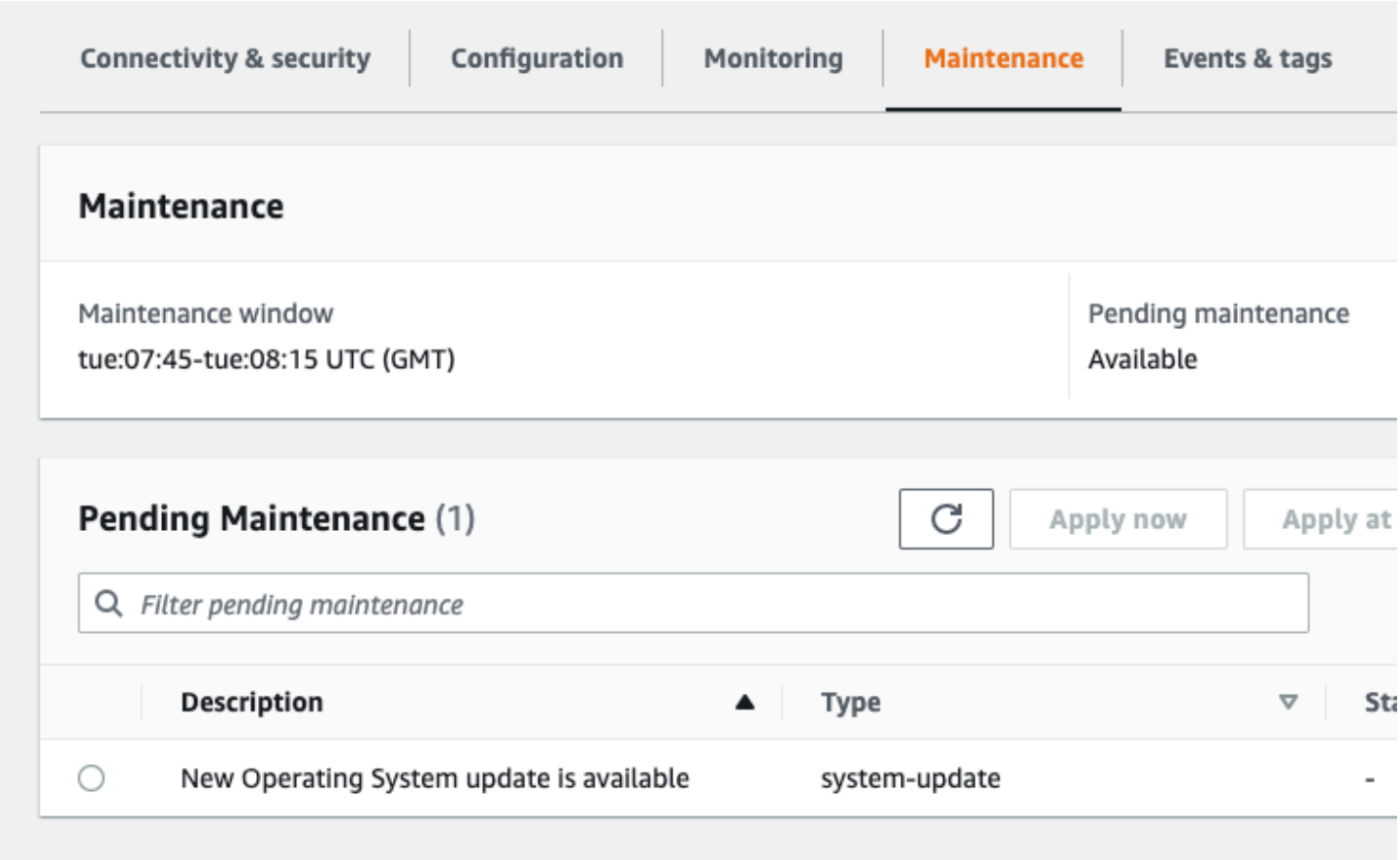
Vous pouvez utiliser la AWS Management Console ou la AWS CLI pour déterminer si une mise à jour est disponible.

Utilisation de AWS Management Console

Pour déterminer si une mise à jour est disponible à l'aide de AWS Management Console :

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home

2. Dans le volet de navigation, choisissez Clusters, puis sélectionnez l'instance.
3. Choisissez Maintenance.
4. Dans la section Maintenance en attente, recherchez la mise à jour du système d'exploitation.



Vous pouvez sélectionner la mise à jour du système d'exploitation et cliquer sur Appliquer maintenant ou sur Appliquer à la fenêtre de maintenance suivante dans la section Maintenance en attente. Si la valeur de maintenance correspond à la fenêtre suivante, reportez les éléments de maintenance en choisissant Différer la mise à niveau. Vous ne pouvez pas reporter une action de maintenance en cours.

Vous pouvez également choisir l'instance dans une liste de clusters en cliquant sur Clusters dans le volet de navigation et en sélectionnant Appliquer maintenant ou Appliquer à la prochaine fenêtre de maintenance dans le menu Actions.

Utilisation de AWS INTERFACE DE LIGNE DE COMMANDE (CLI)

Pour déterminer si une mise à jour est disponible à l'aide du AWS CLI, appelez la `describe-pending-maintenance-actions` commande suivante :

```
aws neptune describe-pending-maintenance-actions
```

```
{
    "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:myneptune",
    "PendingMaintenanceActionDetails": [
        {
            "Action": "system-update",
            "Description": "New Operating System update is available"
        }
    ]
}
```

Pour appliquer les mises à jour du système d'exploitation, appelez la `apply-pending-maintenance-action` commande :

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --resource-identifier (ARN of your DB instance) \
    --opt-in-type immediate
```

Protection des données de votre base de données Amazon Neptune

Le modèle de responsabilité AWS partagée (modèle de) s'applique à la protection des données dans Amazon Neptune. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez la FAQ sur la confidentialité des données et les . Pour plus d'informations sur la protection des données en Europe, consultez le Centre du règlement général sur la protection des données (RGPD).

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity

Center ou Gestion des identités et des accès AWS (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS disponibles, consultez Norme FIPS (Federal Information Processing Standard) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec Neptune ou un autre utilisateur à Services AWS l'aide de la console, de l'API ou AWS des AWS CLI SDK. Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

> **Important**
>
> Le protocole TLS 1.3 n'est pris en charge que pour les versions 1.3.2.0 et ultérieures du moteur Neptune.

Vous utilisez des appels d'API AWS publiés pour gérer Neptune via le réseau. Les clients doivent prendre en charge le protocole TLS (Transport Layer Security) version 1.2 ou ultérieure à l'aide de

suites de chiffrement robustes, comme décrit dans Le chiffrement se connecte aux bases de données Neptune. La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

Les sections suivantes décrivent la façon dont les données Neptune sont protégées.

Rubriques

- Sécurisation de votre base de données Amazon Neptune avec Amazon VPC
- Chiffrer les connexions à votre base de données Amazon Neptune avec SSL/HTTPS
- Chiffrer les ressources Neptune au repos

Sécurisation de votre base de données Amazon Neptune avec Amazon VPC

Un cluster de base de données Amazon Neptune ne peut être créé que dans un Amazon Virtual Private Cloud (Amazon VPC), et ses points de terminaison sont accessibles au sein de ce VPC, généralement à partir d'une instance Amazon Elastic Compute Cloud (Amazon EC2) exécutée dans ce VPC. Il est également possible d'y accéder via un point de terminaison public. Pour plus d'informations sur les points de terminaison publics, consultez la section Points de terminaison publics Neptune.

Vous pouvez sécuriser vos données Neptune en limitant l'accès au VPC sur lequel se trouve le cluster de bases de données Neptune, comme décrit dans Connexion à un cluster Amazon Neptune.

Chiffrer les connexions à votre base de données Amazon Neptune avec SSL/HTTPS

Amazon Neptune autorise uniquement les connexions SSL (Secure Sockets Layer) via HTTPS à n'importe quel point de terminaison d'instance ou de cluster.

Neptune nécessite au moins la version 1.2 de TLS, en utilisant les suites de chiffrement puissantes suivantes :

- `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`

À partir de la version 1.3.2.0 du moteur Neptune, Neptune prend en charge la version 1.3 du TLS en utilisant les suites de chiffrement suivantes :

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384

Même lorsque les connexions HTTP sont autorisées dans les versions antérieures du moteur, tout cluster de bases de données qui utilise un nouveau groupe de paramètres de cluster de bases de données doit utiliser le protocole SSL par défaut. Pour protéger vos données, les points de terminaison Neptune des versions du moteur *`1.0.4.0`* et supérieures ne prennent en charge que les demandes HTTPS. Pour plus d'informations, consultez Utilisation du point de terminaison HTTP REST pour se connecter à une instance de base de données Neptune.

Neptune fournit automatiquement des certificats SSL pour vos instances de base de données Neptune. Vous n'avez pas besoin de demander les certificats. Les certificats sont fournis lors de la création d'une nouvelle instance.

Neptune attribue un seul certificat SSL générique aux instances de votre compte pour chaque région. AWS Le certificat fournit des entrées pour les points de terminaison de cluster, les points de terminaison de cluster en lecture seule et les points de terminaison d'instance.

Détails du certificat

Les entrées suivantes sont incluses dans le certificat fourni :

- Point de terminaison de cluster : `*.cluster-`*`a1b2c3d4wxyz.region`*`.neptune.amazonaws.com`
- Read-only point de terminaison — `*.cluster-ro-`*`a1b2c3d4wxyz.region`*`.neptune.amazonaws.com`
- Point de terminaison d'instance : `*.`*`a1b2c3d4wxyz.region`*`.neptune.amazonaws.com`

Seules les entrées répertoriées ici sont prises en charge.

Connexions proxy

Les certificats prennent uniquement en charge les noms d'hôte qui sont répertoriés dans la section précédente.

Si vous utilisez un équilibreur de charge ou un serveur proxy (par exemple, HAProxy), vous devez utiliser la terminaison SSL et avoir votre propre certificat SSL sur le serveur proxy.

La transmission SSL ne fonctionne pas, car les certificats SSL fournis ne correspondent au nom d'hôte du serveur proxy.

Certificats d'autorité de certification racines

Les certificats pour les instances Neptune sont normalement validés à l'aide du référentiel d'approbations local du système d'exploitation ou du kit SDK (par exemple, le kit SDK Java).

Si vous avez besoin de fournir un certificat racine manuellement, vous pouvez télécharger le certificat d'autorité de certification racine Amazon au format PEM à partir du référentiel de politiques Amazon Trust Services.

En savoir plus

Pour plus d'informations sur la connexion à des points de terminaison Neptune avec SSL, consultez the section called “Installation de la console Gremlin” et the section called “HTTP REST”.

Chiffrer les ressources Neptune au repos

Data-at-rest le chiffrement est la AWS recommandation. Pour plus d'informations, voir Data-at-Restet Data-in-Transit chiffrement. Le chiffrement est appliqué dans la AWS console lorsque vous créez un nouveau cluster de base de données Neptune ou une nouvelle base de données globale Neptune. Cela fournit une couche supplémentaire de protection des données. Il protège vos données contre tout accès non autorisé au stockage sous-jacent et contribue à satisfaire aux exigences de conformité en matière de chiffrement des données au repos.

Pour gérer les clés utilisées pour chiffrer et déchiffrer vos ressources Neptune, vous utilisez ().AWS Key Management ServiceAWS KMS AWS KMS combine du matériel et des logiciels sécurisés et hautement disponibles pour fournir un système de gestion des clés adapté au cloud. À l'aide de AWS KMS, vous pouvez créer des clés de chiffrement et définir les politiques qui contrôlent la manière dont ces clés peuvent être utilisées. AWS KMS prend en charge AWS CloudTrail, afin que vous puissiez auditer l'utilisation des clés pour vérifier que les clés sont utilisées de manière appropriée.

Au repos, tous les journaux, sauvegardes et instantanés associés sont chiffrés pour tous les clusters de base de données Neptune chiffrés. Le chiffrement Neptune ne s'applique pas aux journaux exportés vers Amazon. CloudWatch

Chiffrement des ressources Neptune

Lorsque vous créez un cluster de base de données Neptune ou une base de données globale Neptune, vous pouvez fournir l'identifiant de clé pour votre AWS KMS clé de chiffrement. Si vous ne spécifiez aucun identifiant de AWS KMS clé, Neptune utilise votre clé de chiffrement Amazon RDS par défaut (`aws/rds`) dans la région. AWS KMS crée une clé de chiffrement par défaut pour chaque région de votre AWS compte. Pour un cluster Neptune Global, il contiendra autant de AWS KMS clés que de régions.

Une fois que vous avez créé une ressource Neptune, vous ne pouvez pas modifier la clé de chiffrement de cette ressource. Assurez-vous donc de déterminer vos exigences en matière de clé de chiffrement avant de créer votre ressource Neptune. Si une AWS KMS clé différente est requise, vous pouvez utiliser un instantané du cluster de base de données Neptune existant pour en créer un nouveau avec une AWS KMS clé différente (voirRestauration à partir d'un instantané de cluster de base de données).

Vous pouvez utiliser l'Amazon Resource Name (ARN) d'une clé d'un autre compte pour chiffrer une ressource Neptune. Si vous créez une ressource Neptune avec le même AWS compte qui possède la clé de AWS KMS chiffrement, l' AWS KMS ID de clé que vous transmettez peut être l'alias de la AWS KMS clé au lieu de l'ARN de la clé.

> **Important**
>
> Si Neptune perd l'accès à la clé de chiffrement d'un cluster de base de données Neptune, par exemple lorsque l'accès de Neptune à une clé est révoqué, le cluster chiffré est placé dans un état terminal et ne peut être restauré qu'à partir d'une sauvegarde. Nous vous recommandons vivement de toujours activer les sauvegardes pour les clusters de base de données Neptune chiffrés afin de vous prémunir contre la perte de données chiffrées dans vos bases de données.

Autorisations de clés nécessaires pour activer le chiffrement

L'utilisateur ou le rôle IAM qui crée un cluster de base de données Neptune doit disposer au moins des autorisations suivantes pour la clé KMS :

- `"kms:Encrypt"`
- `"kms:Decrypt"`
- `"kms:GenerateDataKey"`

- "kms:ReEncryptTo"
- "kms:GenerateDataKeyWithoutPlaintext"
- "kms:CreateGrant"
- "kms:ReEncryptFrom"
- "kms:DescribeKey"

Voici un exemple (pour une us-east-1 région) d'une politique clé incluant les autorisations nécessaires :

JSON

```
{
  "Version":"2012-10-17",
  "Id": "key-consolepolicy-3",
  "Statement": [
    {
      "Sid": "Enable Permissions for root principal",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow use of the key for Neptune",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/NeptuneFullAccess"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:ReEncryptTo",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:CreateGrant",
        "kms:ReEncryptFrom",
        "kms:DescribeKey"
      ],
```

```
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "rds.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "Deny use of the key for non Neptune",
      "Effect": "Deny",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/NeptuneFullAccess"
      },
      "Action": [
        "kms:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "kms:ViaService": "rds.us-east-1.amazonaws.com"
        }
      }
    }
  ]
}
```

- La première déclaration de cette politique est facultative. Elle donne accès au principal root de l'utilisateur.
- La deuxième instruction donne accès à toutes les AWS KMS API requises pour ce rôle, jusqu'au principal du service RDS.
- La troisième déclaration renforce davantage la sécurité en indiquant que cette clé n'est pas utilisable par ce rôle pour aucun autre AWS service.

Vous pouvez également réduire davantage les autorisations `createGrant` en ajoutant :

```
"Condition": {
  "Bool": {
    "kms:GrantIsForAWSResource": true
  }
}
```

Limites du chiffrement Neptune

Neptune Encryption présente les limites suivantes :

- Vous ne pouvez pas convertir un cluster de base de données Neptune non chiffré en cluster chiffré. Vous ne pouvez activer le chiffrement pour un cluster de base de données Neptune que lors de sa création. Toutefois, vous pouvez restaurer un instantané de cluster de base de données Neptune non chiffré vers un cluster de base de données Neptune chiffré. Pour ce faire, spécifiez une clé de chiffrement KMS lors de la restauration à partir de l'instantané non chiffré du cluster de base de données Neptune.
- Pour des raisons de compatibilité, il est toujours possible de créer un cluster de base de données Neptune non chiffré via la AWS CLI et les SDK. La console permet uniquement la création de clusters de base de données Neptune chiffrés.
- Vous ne pouvez pas mélanger des clusters de base de données Neptune chiffrés et non chiffrés dans la même base de données globale Neptune. Tous les clusters sont chiffrés ou tous les clusters ne sont pas chiffrés. Cela est appliqué dans la configuration de la base de données globale Neptune.

Authentification de votre base de données Amazon Neptune avec Gestion des identités et des accès AWS

Gestion des identités et des accès AWS (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent les utilisateurs qui peuvent être authentifiés (connectés) et autorisés (via les autorisations appropriées) à utiliser des ressources Neptune. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Vous pouvez utiliser Gestion des identités et des accès AWS (IAM) pour vous authentifier auprès de votre instance de base de données Neptune ou de votre cluster de base de données. Lorsque l'authentification de base de données IAM est activée, chaque demande doit être signée à l'aide de AWS la version 4 de Signature.

AWS Signature Version 4 ajoute des informations d'authentification aux AWS demandes. Pour des raisons de sécurité, toutes les demandes adressées aux clusters de bases de données Neptune avec l'authentification IAM activée doivent être signées avec une clé d'accès. Cette clé comprend un ID de clé d'accès et une clé d'accès secrète. L'authentification est gérée de manière externe à l'aide de politiques IAM.

Neptune s'authentifie lors de la connexion et, pour les WebSockets connexions, vérifie régulièrement les autorisations afin de s'assurer que l'utilisateur y a toujours accès.

> **Note**
>
> - La révocation, la suppression ou la rotation des informations d'identification associées à l'utilisateur IAM n'est pas recommandée, car elle ne met pas fin aux connexions déjà ouvertes.
> - Le nombre de WebSocket connexions simultanées par instance de base de données et la durée pendant laquelle une connexion peut rester ouverte sont limités. Pour de plus amples informations, veuillez consulter WebSockets Limites.

L'utilisation d'IAM dépend de votre rôle

La façon dont vous utilisez Gestion des identités et des accès AWS (IAM) varie en fonction du travail que vous effectuez dans Neptune.

Utilisateur du service : si vous utilisez le service Neptune pour effectuer une tâche, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin pour pouvoir utiliser le plan de données Neptune. Quand vous aurez besoin de davantage d'accès pour faire votre travail, comprendre comment l'accès aux données est géré vous aidera à demander les autorisations appropriées à votre administrateur.

Administrateur du service : si vous êtes chargé des ressources Neptune de votre entreprise, vous avez probablement accès aux actions de gestion Neptune, qui correspondent à l'API de gestion Neptune. Il peut également vous incomber de déterminer les actions d'accès aux données et les ressources dont les utilisateurs du service Neptune ont besoin pour faire leur travail. Un administrateur IAM peut ensuite appliquer des politiques IAM pour modifier les autorisations des utilisateurs de votre service.

Administrateur IAM : si vous êtes administrateur IAM, vous devez écrire des politiques IAM pour gérer à la fois l'accès des données à Neptune et leur gestion. Pour obtenir des exemples de politiques basées sur l'identité Neptune que vous pouvez utiliser, consultez Utilisation de différents types de politique IAM pour contrôler l'accès à Neptune.

Authentification avec des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en tant qu'identité fédérée à l'aide d'informations d'identification provenant d'une source d'identité telle que AWS IAM Identity Center (IAM Identity Center), d'une authentification unique ou d'informations d'identification. Google/Facebook Pour plus d'informations sur la connexion, consultez Connexion à votre Compte AWS dans le Guide de l'utilisateur Connexion à AWS .

Pour l'accès par programmation, AWS fournit un SDK et une CLI pour signer les demandes de manière cryptographique. Pour plus d'informations, consultez Signature AWS Version 4 pour les demandes d'API dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une seule identité de connexion appelée utilisateur Compte AWS root qui dispose d'un accès complet à toutes Services AWS les ressources. Il est vivement déconseillé d'utiliser l'utilisateur racine pour vos tâches quotidiennes. Pour les tâches qui requièrent des informations d'identification de l'utilisateur racine, consultez Tâches qui requièrent les informations d'identification de l'utilisateur racine dans le Guide de l'utilisateur IAM.

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité qui dispose d'autorisations spécifiques pour une seule personne ou application. Nous vous recommandons d'utiliser ces informations d'identification temporaires au lieu des utilisateurs IAM avec des informations d'identification à long terme. Pour plus d'informations, voir Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires dans le guide de l'utilisateur IAM.

Les groupes IAM spécifient une collection d'utilisateurs IAM et permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Pour plus d'informations, consultez Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité dotée d'autorisations spécifiques qui fournit des informations d'identification temporaires. Vous pouvez assumer un rôle en passant d'un rôle d'utilisateur à un rôle

IAM (console) ou en appelant une opération d' AWS API AWS CLI ou d'API. Pour plus d'informations, consultez Méthodes pour endosser un rôle dans le Guide de l'utilisateur IAM.

Les rôles IAM sont utiles pour l'accès des utilisateurs fédérés, les autorisations temporaires des utilisateurs IAM, les accès intercompte, les accès entre services et les applications exécutées sur Amazon EC2. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Activation de l'authentification de base de données IAM dans Amazon Neptune

L'authentification de base de données IAM est désactivée par défaut lorsque vous créez un cluster de bases de données Amazon Neptune. Vous pouvez activer l'authentification de base de données IAM (ou la désactiver à nouveau) à l'aide d' AWS Management Console.

Pour créer un cluster de bases de données Neptune avec une authentification IAM à l'aide de la console, suivez les instructions de création d'un cluster de bases de données Neptune dans Lancement d'un cluster de base de données Neptune à l'aide du AWS Management Console.

Au cours du processus de création, faites défiler l'écran jusqu'à Paramètres supplémentaires et sélectionnez Activer l'authentification IAM.

Pour activer ou désactiver l'authentification IAM pour un cluster ou une instance de base de données existants

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le panneau de navigation, choisissez Clusters.
3. Choisissez le cluster de base de données Neptune que vous souhaitez modifier, puis choisissez Modifier.
4. Faites défiler l'écran vers le bas jusqu'à Paramètres supplémentaires. Pour l'authentification IAM DB, choisissez Activer l'authentification IAM ou Désactiver l'authentification IAM (pour la désactiver). Ensuite, sélectionnez Suivant.
5. Sous Planification des modifications, choisissez quand appliquer la modification, puis choisissez Soumettre.

> **Important**
>
> L'activation ou la désactivation de l'authentification de base de données IAM entraîne le redémarrage du moteur de base de données. Ce redémarrage met fin à toutes les connexions existantes au cluster et entraîne une brève interruption de disponibilité. Prévoyez d'effectuer cette modification pendant une période de maintenance ou à un moment où l'impact d'une interruption de connexion est minime.

Connexion à votre base de données Amazon Neptune à l'aide de Gestion des identités et des accès AWS authentification

Les ressources Amazon Neptune pour lesquelles l'authentification IAM DB est activée nécessitent que toutes les requêtes HTTP soient signées à l'aide de AWS Signature Version 4. Pour obtenir des informations générales sur les demandes de AWS signature avec Signature version 4, consultez la section Signing AWS API requests.

AWS Signature Version 4 est le processus permettant d'ajouter des informations d'authentification aux AWS demandes. Pour des raisons de sécurité, la plupart des demandes AWS doivent être signées avec une clé d'accès, qui consiste en un identifiant de clé d'accès et une clé d'accès secrète.

> **Note**
>
> Si vous utilisez des informations d'identification temporaires, celles-ci expirent après un intervalle spécifique, y compris le jeton de session.
> Vous devez mettre à jour votre jeton de session lorsque vous demandez de nouvelles informations d'identification. Pour plus d'informations, consultez la section Utilisation d'informations d'identification de sécurité temporaires pour demander l'accès aux AWS ressources.

> **Important**
>
> Pour accéder à Neptune avec IAM-based authentification, vous devez créer des requêtes HTTP et les signer vous-même.

Fonctionnement de Signature Version 4

1. Vous créez une demande canonique.
2. Vous utilisez la demande canonique et d'autres informations pour créer une chaîne à signer.
3. Vous utilisez votre clé d'accès AWS secrète pour obtenir une clé de signature, puis vous utilisez cette clé de signature et la chaîne à signer pour créer une signature.
4. Vous ajoutez la signature obtenue à la demande HTTP dans un en-tête ou en tant que paramètre de chaîne de requête.

Lorsque Neptune reçoit la demande, ce service exécute les mêmes étapes que celles que vous avez suivies pour calculer la signature. Neptune compare ensuite la signature calculée à celle que vous avez envoyée avec la demande. Si les signatures correspondent, la demande est traitée. Si les signatures ne correspondent pas, la demande est rejetée.

Pour des informations générales sur les demandes de AWS signature avec la version 4 de Signature, voir Processus de signature de la version 4 dans le Références générales AWS.

Les sections suivantes contiennent des exemples qui illustrent l'envoi de demandes signées aux points de terminaison Gremlin et SPARQL d'une instance de base de données Neptune avec l'authentification IAM activée.

Rubriques

- Conditions requises pour connecter les bases de données Amazon Neptune à l'aide de l'authentification IAM
- Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM depuis la ligne de commande
- Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec la console Gremlin
- Connexion aux bases de données Amazon Neptune à l'aide d'IAM avec Gremlin Java
- Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Java et SPARQL
- Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec SPARQL et Node.js
- Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Python
- Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Gremlin Python

- Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Gremlin JavaScript
- Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Gremlin Go
- Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Gremlin .NET

Conditions requises pour connecter les bases de données Amazon Neptune à l'aide de l'authentification IAM

Voici les instructions d'installation d'Apache Maven et de Java 8 sur une instance Amazon EC2. Elles sont nécessaires pour les exemples d'authentification avec Amazon Neptune Signature Version 4.

Pour installer Apache Maven et Java 8 sur votre instance EC2

1. Connectez-vous à votre instance Amazon EC2 avec un client SSH.
2. Installez Apache Maven sur votre instance EC2. Si vous utilisez Amazon Linux 2023 (de préférence), utilisez :

   ```
   sudo dnf update -y
   sudo dnf install maven -y
   ```

   Si vous utilisez Amazon Linux 2, téléchargez le dernier fichier binaire à l'adresse suivante https://maven.apache.org/download.cgi:

   ```
   sudo yum remove maven -y
   wget https://dlcdn.apache.org/maven/maven-3/ <version>/binaries/apache-maven-
   <version>-bin.tar.gz
   sudo tar -xzf apache-maven-<version>-bin.tar.gz -C /opt/
   sudo ln -sf /opt/apache-maven-<version> /opt/maven
   echo 'export MAVEN_HOME=/opt/maven' >> ~/.bashrc
   echo 'export PATH=$MAVEN_HOME/bin:$PATH' >> ~/.bashrc
   source ~/.bashrc
   ```

3. Les bibliothèques Gremlin nécessitent Java 8. Entrez ce qui suit pour installer Java 8 sur votre instance EC2.

   ```
   sudo yum install java-1.8.0-devel
   ```

4. Entrez la commande suivante pour définir Java 8 en tant qu'exécution par défaut sur votre instance EC2.

```
sudo /usr/sbin/alternatives --config java
```

   Lorsque vous y êtes invité, saisissez le nombre pour Java 8.

5. Saisissez la commande suivante pour définir Java 8 en tant que compilateur par défaut de votre instance EC2.

```
sudo /usr/sbin/alternatives --config javac
```

   Lorsque vous y êtes invité, saisissez le nombre pour Java 8.

Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM depuis la ligne de commande

Il est particulièrement utile d'avoir recours à un outil en ligne de commande pour envoyer des requêtes à votre cluster de bases de données Neptune, comme l'illustrent de nombreux exemples dans cette documentation. L'outil curl est une excellente option pour communiquer avec les points de terminaison Neptune si l'authentification IAM n'est pas activée.

Toutefois, pour garantir la sécurité de vos données, il est préférable d'activer l'authentification IAM.

Lorsque l'authentification de base de données IAM est activée, chaque demande doit être signée à l'aide de Signature Version 4 (Sig4). L'outil de ligne de commande tiers awscurl utilise la même syntaxe que `curl` et peut signer les requêtes à l'aide de la signature Sig4. La section Utilisation de `awscurl` ci-dessous explique comment utiliser `awscurl` en toute sécurité avec des informations d'identification temporaires.

Configuration d'un outil de ligne de commande pour utiliser le protocole HTTPS

Neptune exige que toutes les connexions utilisent le protocole HTTPS. Tout outil de ligne de commande comme `curl` ou `awscurl` doit accéder aux certificats appropriés pour pouvoir utiliser le protocole HTTPS. Dans la mesure où `curl` ou `awscurl` peut localiser les certificats appropriés, il gère les connexions HTTPS comme des connexions HTTP, sans paramètres supplémentaires. Les exemples de cette documentation sont basés sur ce scénario.

Pour découvrir comment obtenir ces certificats et comment les mettre correctement en forme dans un magasin de certificats CA utilisable par `curl`, consultez SSL Certificate Verification (Vérification des certificats SSL) dans la documentation `curl`.

Vous pouvez ensuite spécifier l'emplacement de ce magasin de certificats CA à l'aide de la variable d'environnement `CURL_CA_BUNDLE`. Sous Windows, `curl` le recherche automatiquement dans un fichier nommé `curl-ca-bundle.crt`. Il examine d'abord dans le même répertoire que `curl.exe`, puis ailleurs sur le chemin. Pour plus d'informations, consultez SSL Certificate Verification (Vérification des certificats SSL).

Utiliser **`awscurl`** avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée

L'outil awscurl utilise la même syntaxe que `curl`, mais nécessite également des informations supplémentaires :

- **`--access_key`** : clé d'accès valide. Si cet élément n'est pas fourni à l'aide de ce paramètre, il doit être fourni dans la variable d'environnement `AWS_ACCESS_KEY_ID` ou dans un fichier de configuration.
- **`--secret_key`** : clé secrète valide qui correspond à la clé d'accès. Si cet élément n'est pas fourni à l'aide de ce paramètre, il doit être fourni dans la variable d'environnement `AWS_SECRET_ACCESS_KEY` ou dans un fichier de configuration.
- **`--security_token`** : jeton de session valide. Si cet élément n'est pas fourni à l'aide de ce paramètre, il doit être fourni dans la variable d'environnement `AWS_SECURITY_TOKEN` ou dans un fichier de configuration.

Auparavant, il était courant d'utiliser des informations d'identification persistantes avec `awscurl`, telles que les informations d'identification utilisateur IAM ou même les informations d'identification root, mais cela n'est pas recommandé. Générez plutôt des informations d'identification temporaires à l'aide de l'une des API Security Token Service (STS) ou de l'un de leurs AWS CLI wrappers AWS.

Il est préférable de placer les valeurs `AccessKeyId`, `SecretAccessKey` et `SessionToken` renvoyées par l'appel STS dans les variables d'environnement appropriées de la session shell plutôt que dans un fichier de configuration. Lorsque le shell se terminera, les informations d'identification seront automatiquement supprimées, ce qui n'est pas le cas avec un fichier de configuration. De même, ne demandez pas une durée plus longue que celle dont vous aurez probablement besoin pour les informations d'identification temporaires.

L'exemple suivant présente les étapes que vous pouvez suivre dans un shell Linux pour obtenir des informations d'identification temporaires valables pendant une demi-heure à l'aide de sts assume-role, puis pour les placer dans des variables d'environnement où `awscurl` peux les trouver :

```
aws sts assume-role \
    --duration-seconds 1800 \
    --role-arn "arn:aws:iam::(account-id):role/(rolename)" \
    --role-session-name AWSCLI-Session > $output
AccessKeyId=$(echo $output | jq '.Credentials''.AccessKeyId')
SecretAccessKey=$(echo $output | jq '.Credentials''.SecretAccessKey')
SessionToken=$(echo $output | jq '.Credentials''.SessionToken')

export AWS_ACCESS_KEY_ID=$AccessKeyId
export AWS_SECRET_ACCESS_KEY=$SecretAccessKey
export AWS_SESSION_TOKEN=$SessionToken
```

Vous pouvez ensuite utiliser `awscurl` pour envoyer une demande signée au cluster de bases de données comme suit :

```
awscurl (your cluster endpoint):8182/status \
    --region us-east-1 \
    --service neptune-db
```

Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec la console Gremlin

Pour vous connecter à Amazon Neptune à l'aide de la console Gremlin avec authentification Signature Version 4, vous devez `requestInterceptor()` connecter un signataire SigV4 à la connexion établie par la commande. `:remote` Cela nécessite que vous configuriez l'`Cluster`objet manuellement, puis que vous le transmettiez à la `:remote` commande.

Notez que cela est très différent de la situation typique où la commande `:remote` utilise un fichier de configuration pour établir la connexion. L'approche utilisant le fichier de configuration ne fonctionne pas, car `requestInterceptor()` doit être défini par programmation et ne peut pas charger sa configuration à partir d'un fichier.

> **Note**
>
> Les exemples suivants utilisent`requestInterceptor()`, qui a été introduit dans la version TinkerPop 3.6.6. Si vous utilisez une TinkerPop version antérieure à 3.6.6 (mais 3.5.5 ou

supérieure), utilisez `handshakeInterceptor()` plutôt que `requestInterceptor()` dans les exemples de code ci-dessous.

Les prérequis suivants sont nécessaires :

- Vous devez fournir les informations d'identification IAM nécessaires pour signer les demandes. Consultez la section Utilisation de la chaîne de fournisseurs d'informations d'identification par défaut dans le Guide du AWS SDK pour Java développeur.
- Vous devez avoir installé une version de la console Gremlin compatible avec la version du moteur Neptune utilisée par le cluster de bases de données.

Si vous utilisez des informations d'identification temporaires, elles expirent après un intervalle spécifié, tout comme le jeton de session. Vous devez donc mettre à jour le jeton de session lorsque vous demandez de nouvelles informations d'identification. Consultez la section Utilisation d'informations d'identification de sécurité temporaires pour demander l'accès aux AWS ressources dans le guide de l'utilisateur IAM.

Pour obtenir de l'aide concernant la connexion à l'aide SSL/TLS, voirSSL/TLS configuration.

Connect la console G705 avec la signature Sig4

1. Démarrez la console Gremlin :

   ```
   $ bin/gremlin.sh
   ```

2. À l'invite `gremlin>`, installez la bibliothèque `amazon-neptune-sigv4-signer` (cette opération ne doit être effectuée qu'une seule fois pour la console) :

   ```
   :install com.amazonaws amazon-neptune-sigv4-signer 2.4.0
   ```

   Si vous rencontrez des problèmes lors de cette étape, il peut être utile de consulter la TinkerPop documentation relative à la configuration de Grape.

> **Note**
>
> Si vous utilisez un proxy HTTP, vous risquez de rencontrer des erreurs lors de cette étape où la commande `:install` ne s'exécute pas. Afin de résoudre ce problème, exécutez les commandes suivantes pour informer la console de l'existence du proxy :
>
> ```
> System.setProperty("https.proxyHost", "(the proxy IP address)")
> System.setProperty("https.proxyPort", "(the proxy port)")
> ```

3. Importez la classe requise pour gérer la signature dans `requestInterceptor()` :

```
:import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider
:import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer
```

4. Si vous utilisez des informations d'identification temporaires, vous devrez également fournir votre jeton de session comme suit :

```
System.setProperty("aws.sessionToken","(your session token)")
```

5. Si vous n'avez pas encore établi les informations d'identification de votre compte, vous pouvez les attribuer comme suit :

```
System.setProperty("aws.accessKeyId","(your access key)")
System.setProperty("aws.secretKey","(your secret key)")
```

6. Construisez manuellement l'objet `Cluster` à connecter à Neptune :

```
cluster = Cluster.build("(host name)")  \
                .enableSsl(true) \
                .requestInterceptor { r ->  \
                  def sigV4Signer = new NeptuneNettyHttpSigV4Signer("(Amazon
 region)", \
                                        DefaultCredentialsProvider.create()); \
                  sigV4Signer.signRequest(r); \
                  return r; } \
                .create()
```

Pour découvrir comment trouver le nom d'hôte de l'instance de base de données Neptune, consultez Connexion aux points de terminaison Amazon Neptune.

7. Pour établir la connexion `:remote`, utilisez le nom de variable de l'objet `Cluster` à l'étape précédente :

```
:remote connect tinkerpop.server cluster
```

8. Entrez la commande suivante pour passer en mode distant. Toutes les requêtes Gremlin sont alors envoyées à la connexion distante.

```
:remote console
```

Connexion aux bases de données Amazon Neptune à l'aide d'IAM avec Gremlin Java

Voici un exemple de connexion à Neptune à l'aide de l'API Java Gkremlin avec signature Sig4 (cela suppose des connaissances générales sur l'utilisation de Maven). Cet exemple utilise la bibliothèque Amazon Neptune SigV4 Signer pour faciliter la signature des demandes. Définissez d'abord les dépendances dans le cadre du fichier `pom.xml` :

> **Note**
>
> Les exemples suivants utilisent`requestInterceptor()`, qui a été introduit dans la version TinkerPop 3.6.6. Si vous utilisez une TinkerPop version antérieure à 3.6.6 (mais 3.5.5 ou supérieure), utilisez `handshakeInterceptor()` plutôt que `requestInterceptor()` dans les exemples de code ci-dessous.

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>amazon-neptune-sigv4-signer</artifactId>
  <version>3.1.0</version>
</dependency>
```

Amazon Neptune SigV4 Signer prend en charge les versions 1.x et 2.x du SDK Java.
AWS Les exemples suivants utilisent 2.x, où `DefaultCredentialsProvider` est une

`software.amazon.awssdk.auth.credentials.AwsCredentialsProvider` instance. Si vous passez de la version 1.x à la version 2.x, consultez les modifications apportées au fournisseur d'informations d'identification dans la documentation du AWS SDK for Java 2.x.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer;
import com.amazonaws.neptune.auth.NeptuneSigV4SignerException;

 ...

System.setProperty("aws.accessKeyId","your-access-key");
System.setProperty("aws.secretKey","your-secret-key");

 ...

Cluster cluster = Cluster.build((your cluster))
                 .enableSsl(true)
                 .requestInterceptor( r ->
                  {
                    try {
                      NeptuneNettyHttpSigV4Signer sigV4Signer =
                        new NeptuneNettyHttpSigV4Signer("(your region)",
 DefaultCredentialsProvider.create());
                      sigV4Signer.signRequest(r);
                    } catch (NeptuneSigV4SignerException e) {
                      throw new RuntimeException("Exception occurred while signing the
 request", e);
                    }
                    return r;
                  }
                 ).create();
try {
  Client client = cluster.connect();
  client.submit("g.V().has('code','IAD')").all().get();
} catch (Exception e) {
  throw new RuntimeException("Exception occurred while connecting to cluster", e);
}
```

Authentification IAM entre comptes

Amazon Neptune prend en charge l'authentification IAM entre comptes grâce à l'utilisation de l'hypothèse de rôles, parfois appelée chaînage de rôles. Pour donner accès à un cluster Neptune à partir d'une application hébergée sur un autre AWS compte :

- Créez un nouvel utilisateur ou rôle IAM dans le AWS compte d'application, avec une politique de confiance qui permet à l'utilisateur ou au rôle d'assumer un autre rôle IAM. Attribuez ce rôle au calcul hébergeant l'application (instance EC2, fonction Lambda, tâche ECS, etc.).

  JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "assumeRolePolicy",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::111122223333:role/role-name"
    }
  ]
}
```

- Créez un nouveau rôle IAM dans le compte de base de données Neptune qui autorise l'accès à la AWS base de données Neptune et autorise l'attribution de rôles depuis le compte d'application IAM. user/role Utilisez une politique de confiance basée sur :

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "(ARN of application account IAM user or role)"
                ]
            },
            "Action": "sts:AssumeRole",
```

```
            "Condition": {}
        }
    ]
}
```

- Utilisez l'exemple de code suivant pour savoir comment utiliser ces deux rôles pour autoriser l'application à accéder à Neptune. Dans cet exemple, le rôle du compte d'application sera assumé via le DefaultCredentialProviderChainlors de la création du`STSclient`. `STSclient`Il est ensuite utilisé via le `STSAssumeRoleSessionCredentialsProvider` pour assumer le rôle hébergé dans le compte de base AWS de données Neptune.

```
public static void main( String[] args )
  {

    /*
     * Establish an STS client from the application account.
     */
    AWSSecurityTokenService client = AWSSecurityTokenServiceClientBuilder
        .standard()
        .build();

    /*
     * Define the role ARN that you will be assuming in the database account where
 the Neptune cluster resides.
     */
    String roleArnToAssume = "arn:aws:iam::012345678901:role/
CrossAccountNeptuneRole";
    String crossAccountSessionName = "cross-account-session-" + UUID.randomUUID();

    /*
     * Change the Credentials Provider in the SigV4 Signer to use the STSAssumeRole
 Provider and provide it
     * with both the role to be assumed, the original STS client, and a session name
 (which can be
     * arbitrary.)
     */
    Cluster cluster = Cluster.build()
                .addContactPoint("neptune-cluster.us-west-2.neptune.amazonaws.com")
                .enableSsl(true)
                .port(8182)
                .requestInterceptor( r ->
                 {
                   try {
```

```
                    NeptuneNettyHttpSigV4Signer sigV4Signer =
                      // new NeptuneNettyHttpSigV4Signer("us-west-2",
DefaultCredentialsProvider.create());
                      new NeptuneNettyHttpSigV4Signer(
                              "us-west-2",
                               new STSAssumeRoleSessionCredentialsProvider
                                  .Builder(roleArnToAssume,
crossAccountSessionName)
                                      .withStsClient(client)
                                      .build());
                    sigV4Signer.signRequest(r);
                  } catch (NeptuneSigV4SignerException e) {
                    throw new RuntimeException("Exception occurred while signing
the request", e);
                  }
                  return r;
                }
               ).create();

   GraphTraversalSource g =
traversal().withRemote(DriverRemoteConnection.using(cluster));

   /* whatever application code is necessary */

   cluster.close();
 }
```

Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Java et SPARQL

Cette section vous montre comment vous connecter à Neptune à l'aide de RDF4J ou d'Apache Jena avec l'authentification Signature Version 4.

Conditions préalables

- Java 8 ou une version ultérieure.
- Apache Maven 3.3 ou version ultérieure.

  Pour plus d'informations sur l'installation de ces logiciels prérequis sur une instance EC2 exécutant Amazon Linux, consultez Conditions requises pour connecter les bases de données Amazon Neptune à l'aide de l'authentification IAM.

- Informations d'identification IAM pour signer les demandes. Pour plus d'informations, consultez Utilisation de la chaîne de fournisseur d'informations d'identification par défaut dans le Manuel du développeur AWS SDK pour Java .

  > **Note**
  >
  > Si vous utilisez des informations d'identification temporaires, celles-ci expirent après un intervalle spécifique, y compris le jeton de session.
  > Vous devez mettre à jour votre jeton de session lorsque vous demandez de nouvelles informations d'identification. Pour plus d'informations, consultez la section Utilisation d'informations d'identification de sécurité temporaires pour demander l'accès aux AWS ressources dans le guide de l'utilisateur IAM.

- Définissez la variable `SERVICE_REGION` sur l'une des valeurs suivantes, en indiquant la région de votre instance de base de données Neptune :
  - USA Est (Virginie du Nord) : `us-east-1`
  - USA Est (Ohio) : `us-east-2`
  - USA Ouest (Californie du Nord) : `us-west-1`
  - USA Ouest (Oregon) : `us-west-2`
  - Canada (Centre) : `ca-central-1`
  - Canada Ouest (Calgary) : `ca-west-1`
  - Amérique du Sud (São Paulo) : `sa-east-1`
  - Europe (Stockholm) : `eu-north-1`
  - Europe (Espagne) : `eu-south-2`
  - Europe (Irlande) : `eu-west-1`
  - Europe (Londres) : `eu-west-2`
  - Europe (Paris) : `eu-west-3`
  - Europe (Francfort) : `eu-central-1`
  - Moyen-Orient (Bahreïn) : `me-south-1`
  - Moyen-Orient (EAU) : `me-central-1`
  - Israël (Tel Aviv) : `il-central-1`
  - Afrique (Le Cap) : `af-south-1`
  - Asie-Pacifique (Hong Kong) : `ap-east-1`

- Asie-Pacifique (Tokyo) : `ap-northeast-1`
- Asie-Pacifique (Séoul) : `ap-northeast-2`
- Asie-Pacifique (Osaka) : `ap-northeast-3`
- Asie-Pacifique (Singapour) : `ap-southeast-1`
- Asie-Pacifique (Sydney) : `ap-southeast-2`
- Asie-Pacifique (Jakarta) : `ap-southeast-3`
- Asie-Pacifique (Melbourne) : `ap-southeast-4`
- Asie-Pacifique (Malaisie) : `ap-southeast-5`
- Asie-Pacifique (Mumbai) : `ap-south-1`
- Asie-Pacifique (Hyderabad) : `ap-south-2`
- Chine (Beijing) : `cn-north-1`
- Chine (Ningxia) : `cn-northwest-1`
- AWS GovCloud (US-West): `us-gov-west-1`
- AWS GovCloud (US-East): `us-gov-east-1`

Pour se connecter à Neptune à l'aide de RDF4J ou d'Apache Jena avec la signature Signature Version 4

1. Clonez le référentiel d'échantillons à partir de GitHub.

   ```
   git clone https://github.com/aws/amazon-neptune-sparql-java-sigv4.git
   ```

2. Accédez au répertoire cloné.

   ```
   cd amazon-neptune-sparql-java-sigv4
   ```

3. Obtenez la dernière version du projet en vérifiant la branche avec la dernière balise.

   ```
   git checkout $(git describe --tags `git rev-list --tags --max-count=1`)
   ```

4. Saisissez l'une des commandes suivantes pour compiler et exécuter l'exemple de code.

   *`your-neptune-endpoint`*Remplacez-le par le nom d'hôte ou l'adresse IP de votre instance de base de données Neptune. La valeur par défaut du port est 8182.

> **Note**
>
> Consultez la section Connexion aux points de terminaison Amazon Neptune pour découvrir comment trouver le nom d'hôte de votre instance de base de données Neptune.

Eclipse RDF4J

Saisissez la commande suivante pour exécuter l'exemple RDF4J.

```
mvn compile exec:java \
    -Dexec.mainClass="com.amazonaws.neptune.client.rdf4j.NeptuneRdf4JSigV4Example"
 \
    -Dexec.args="https://your-neptune-endpoint:portsparql region-name"
```

Apache Jena

Saisissez la commande suivante pour exécuter l'exemple Apache Jena.

```
mvn compile exec:java \
    -Dexec.mainClass="com.amazonaws.neptune.client.jena.NeptuneJenaSigV4Example" \
    -Dexec.args="https://your-neptune-endpoint:port"
```

5. Pour afficher le code source de l'exemple, consultez les exemples du répertoire `src/main/java/com/amazonaws/neptune/client/`.

Pour utiliser le pilote de signature SigV4 dans votre propre application Java, ajoutez le package Maven `amazon-neptune-sigv4-signer` à la section `<dependencies>` de votre fichier `pom.xml`. Nous vous recommandons d'utiliser les exemples comme point de départ.

Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec SPARQL et Node.js

Interrogation à l'aide de la signature Signature V4 et du AWS SDK pour Javascript V3

Voici un exemple de connexion à Neptune SPARQL à l'aide Node.js de l'authentification Signature Version 4 et du AWS SDK pour Javascript V3 :

```
const { HttpRequest }  = require('@smithy/protocol-http');
const { fromNodeProviderChain } = require('@aws-sdk/credential-providers');
const { SignatureV4 } = require('@smithy/signature-v4');
const { Sha256 } = require('@aws-crypto/sha256-universal');
const https = require('https');

var region = 'us-west-2'; // e.g. us-west-1
var neptune_endpoint = 'your-Neptune-cluster-endpoint';  // like: 'cluster-
id.region.neptune.amazonaws.com'
var query = `query=PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX class: <http://aws.amazon.com/neptune/csv2rdf/class/>
PREFIX resource: <http://aws.amazon.com/neptune/csv2rdf/resource/>
PREFIX prop: <http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/>
PREFIX objprop: <http://aws.amazon.com/neptune/csv2rdf/objectProperty/>

SELECT ?movies ?title WHERE {
  ?jel prop:name "James Earl Jones" .
  ?movies ?p2 ?jel .
  ?movies prop:title ?title
} LIMIT 10`;

runQuery(query);

function runQuery(q) {
  var request = new HttpRequest({
    hostname: neptune_endpoint,
    port: 8182,
    path: 'sparql',
    body: encodeURI(query),
    headers: {
      'Content-Type': 'application/x-www-form-urlencoded',
      'host': neptune_endpoint + ':8182',
    },
    method: 'POST',
  });

  const credentialProvider = fromNodeProviderChain();
  let credentials = credentialProvider();
  credentials.then(
    (cred)=>{
      var signer = new SignatureV4({credentials: cred, region: region, sha256: Sha256,
 service: 'neptune-db'});
      signer.sign(request).then(
```

```
        (req)=>{
          var responseBody = '';
          var sendreq = https.request(
            {
              host: req.hostname,
              port: req.port,
              path: req.path,
              method: req.method,
              headers: req.headers,
            },
          (res) => {
            res.on('data', (chunk) => { responseBody += chunk; });
            res.on('end', () => {
                console.log(JSON.parse(responseBody));
            });
          });
          sendreq.write(req.body);
          sendreq.end();
        }
      );
    },
    (err)=>{
      console.error(err);
    }
  );
}
```

Interrogation à l'aide de la signature Signature V4 et du AWS SDK pour Javascript V2

Voici un exemple de connexion à Neptune SPARQL à l'aide Node.js de l'authentification Signature Version 4 et du AWS SDK pour Javascript V2 :

```
var AWS = require('aws-sdk');

var region = 'us-west-2'; // e.g. us-west-1
var neptune_endpoint = 'your-Neptune-cluster-endpoint';  // like: 'cluster-
id.region.neptune.amazonaws.com'
var query = `query=PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX class: <http://aws.amazon.com/neptune/csv2rdf/class/>
PREFIX resource: <http://aws.amazon.com/neptune/csv2rdf/resource/>
PREFIX prop: <http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/>
PREFIX objprop: <http://aws.amazon.com/neptune/csv2rdf/objectProperty/>
```

```
SELECT ?movies ?title WHERE {
    ?jel prop:name "James Earl Jones" .
    ?movies ?p2 ?jel .
    ?movies prop:title ?title
} LIMIT 10`;

runQuery(query);

function runQuery(q) {

    var endpoint = new AWS.Endpoint(neptune_endpoint);
    endpoint.port = 8182;
    var request = new AWS.HttpRequest(endpoint, region);
    request.path += 'sparql';
    request.body = encodeURI(query);
    request.headers['Content-Type'] = 'application/x-www-form-urlencoded';
    request.headers['host'] = neptune_endpoint;
    request.method = 'POST';

    var credentials = new AWS.CredentialProviderChain();
    credentials.resolve((err, cred)=>{
        var signer = new AWS.Signers.V4(request, 'neptune-db');
        signer.addAuthorization(cred, new Date());
    });

    var client = new AWS.HttpClient();
    client.handleRequest(request, null, function(response) {
        console.log(response.statusCode + ' ' + response.statusMessage);
        var responseBody = '';
        response.on('data', function (chunk) {
            responseBody += chunk;
        });
        response.on('end', function (chunk) {
            console.log('Response body: ' + responseBody);
        });
    }, function(error) {
        console.log('Error: ' + error);
    });
}
```

Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Python

Le `boto3` `neptunedata` client fournit le moyen le plus simple de se connecter à une base de données IAM-enabled Neptune à partir de Python. Le client gère automatiquement la signature de Signature Version 4, vous n'avez donc pas besoin de signer vous-même les demandes.

Conditions préalables

- Python 3.x
- La `boto3` bibliothèque : `pip install boto3`
- AWS informations d'identification configurées via n'importe quelle méthode standard (variables d'environnement, fichier de AWS configuration, profil d'instance ou rôle d'exécution Lambda)
- Une politique IAM qui autorise `neptune-db:*` des actions sur votre cluster Neptune

Connexion avec les informations d'identification par défaut

Cette approche fonctionne lorsqu'il est `boto3` possible de résoudre automatiquement les informations d'identification, par exemple sur les instances Amazon Elastic Compute Cloud avec des profils d'instance, dans des AWS Lambda fonctions ou avec AWS des profils configurés.

Remplacez l'URL du point de terminaison par le point de terminaison de votre cluster Neptune.

Gremlin et OpenCypher

```
import boto3
from botocore.config import Config

cfg = Config(retries={"total_max_attempts": 1, "mode": "standard"}, read_timeout=None)

neptune = boto3.client('neptunedata', config=cfg,
    region_name='us-east-1',
    endpoint_url='https://your-neptune-endpoint:8182')

# Gremlin
resp = neptune.execute_gremlin_query(gremlinQuery='g.V().limit(1)')
print(resp['result'])

# openCypher
resp = neptune.execute_open_cypher_query(openCypherQuery='MATCH (n) RETURN n LIMIT 1')
```

```
print(resp['results'])
```

SPARQL

Le `boto3` `neptunedata` client ne prend actuellement pas en charge le SPARQL. Pour envoyer des requêtes SPARQL à IAM-enabled une base de données Neptune, signez les demandes manuellement à l'aide de Signature Version 4 avec `botocore` les utilitaires de signature de demandes et `requests` la bibliothèque (). `pip install boto3 requests`

```
import requests
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from botocore.session import Session

endpoint = 'https://your-neptune-endpoint:8182'
region = 'us-east-1'

query = 'SELECT ?s ?p ?o WHERE { ?s ?p ?o } LIMIT 1'

request = AWSRequest(method='POST', url=f'{endpoint}/sparql/',
    data={'query': query})
SigV4Auth(Session().get_credentials(), 'neptune-db', region).add_auth(request)

resp = requests.post(f'{endpoint}/sparql/', headers=request.headers,
    data={'query': query})
print(resp.json())
```

> **Note**
>
> Cet exemple permet `botocore.session.Session` de résoudre automatiquement les AWS informations d'identification à partir de variables d'environnement, de fichiers de AWS configuration, de profils d'instance ou de rôles d'exécution Lambda. Il n'est pas nécessaire de définir les informations d'identification de manière explicite.

Utilisation d'informations d'identification temporaires

Pour un accès entre comptes, des sessions limitées dans le temps ou comme meilleure pratique de sécurité afin d'éviter des identifiants de longue durée, utilisez-le AWS STS pour obtenir des informations d'identification temporaires et créer un. `boto3.Session`

Gremlin et OpenCypher

```
import boto3
from botocore.config import Config

sts = boto3.client('sts')
creds = sts.assume_role(
    RoleArn='arn:aws:iam::123456789012:role/NeptuneRole',
    RoleSessionName='neptune-session'
)['Credentials']

session = boto3.Session(
    aws_access_key_id=creds['AccessKeyId'],
    aws_secret_access_key=creds['SecretAccessKey'],
    aws_session_token=creds['SessionToken'],
    region_name='us-east-1'
)

cfg = Config(retries={"total_max_attempts": 1, "mode": "standard"}, read_timeout=None)

neptune = session.client('neptunedata', config=cfg,
    endpoint_url='https://your-neptune-endpoint:8182')

# Gremlin
resp = neptune.execute_gremlin_query(gremlinQuery='g.V().limit(1)')
print(resp['result'])

# openCypher
resp = neptune.execute_open_cypher_query(openCypherQuery='MATCH (n) RETURN n LIMIT 1')
print(resp['results'])
```

> **Note**
>
> Les informations d'identification temporaires expirent après un intervalle spécifié (la valeur par défaut est de 1 heure) et ne sont pas automatiquement actualisées par le client. Pour les applications de longue durée, utilisez plutôt des informations d'identification basées sur des profils ou des profils d'instance.

SPARQL

Étant donné que le `boto3 neptunedata` client ne prend pas actuellement en charge le SPARQL, vous devez signer les demandes manuellement à l'aide des informations d'identification temporaires.

```
import boto3
import requests
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import ReadOnlyCredentials

sts = boto3.client('sts')
creds = sts.assume_role(
    RoleArn='arn:aws:iam::123456789012:role/NeptuneRole',
    RoleSessionName='neptune-session'
)['Credentials']

endpoint = 'https://your-neptune-endpoint:8182'
region = 'us-east-1'

query = 'SELECT ?s ?p ?o WHERE { ?s ?p ?o } LIMIT 1'

request = AWSRequest(method='POST', url=f'{endpoint}/sparql/',
    data={'query': query})
SigV4Auth(ReadOnlyCredentials(
    creds['AccessKeyId'], creds['SecretAccessKey'], creds['SessionToken']
), 'neptune-db', region).add_auth(request)

resp = requests.post(f'{endpoint}/sparql/', headers=request.headers,
    data={'query': query})
print(resp.json())
```

Utilisation avec AWS Lambda

Dans Lambda, le rôle d'exécution fournit automatiquement des informations d'identification via. `boto3` L'exemple d'informations d'identification par défaut fonctionne sans modification.

Pour un exemple Lambda complet avec gestion des connexions et logique de nouvelle tentative, voir. the section called “Exemples de fonctions Lambda”

Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Gremlin Python

Présentation de

Ce guide explique comment se connecter à une base de données Amazon Neptune avec l'authentification IAM activée à l'aide du pilote Python Gremlin, avec l'authentification Signature Version 4 et le AWS SDK for Python (Boto3).

Création d'une connexion de base

Utilisez l'exemple de code suivant pour savoir comment établir une connexion de base avec l'authentification IAM à l'aide du pilote Python Garmlin.

```
from boto3 import Session
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest

from gremlin_python.process.anonymous_traversal import traversal
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection

def main():
    endpoint = 'your.cluster.endpoint.neptune.amazonaws.com'
    conn_string = 'wss://' + endpoint + ':8182/gremlin'
    default_region = 'us-east-1'
    service = 'neptune-db'

    credentials = Session().get_credentials()
    if credentials is None:
        raise Exception("No AWS credentials found")
    creds = credentials.get_frozen_credentials()
    # region set inside config profile or via AWS_DEFAULT_REGION environment variable
 will be loaded
    region = Session().region_name if Session().region_name else default_region

    request = AWSRequest(method='GET', url=conn_string, data=None)
    SigV4Auth(creds, service, region).add_auth(request)

    rc = DriverRemoteConnection(conn_string, 'g', headers=request.headers.items())
    g = traversal().with_remote(rc)

    # simple query to verify connection
```

```
    count = g.V().count().next()
    print('Vertex count: ' + str(count))

    # cleanup
    rc.close()

if __name__ == "__main__":
    main()
```

Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Gremlin JavaScript

Présentation de

Ce guide explique comment se connecter à une base de données Amazon Neptune avec l'authentification IAM activée à l'aide du JavaScript pilote G705, avec l'authentification Signature version 4 et le AWS SDK pour Javascript v3.

Conditions préalables

- Un cluster Amazon Neptune sur lequel l'authentification IAM est activée.
- Node 13 ou version ultérieure (reportez-vous aux versions minimales pour Gremlin JavaScript et au AWS SDK pour Javascript v3).
- AWS informations d'identification configurées (via des variables d'environnement, un fichier d'informations d'identification partagé ou un rôle IAM).

Création d'une connexion de base

Utilisez l'exemple de code suivant pour savoir comment établir une connexion de base avec l'authentification IAM à l'aide du pilote Gkremlin. JavaScript

```
const { fromNodeProviderChain } = require('@aws-sdk/credential-providers');
const { getUrlAndHeaders } = require('gremlin-aws-sigv4/lib/utils');
const { loadConfig } = require("@smithy/node-config-provider");
const { NODE_REGION_CONFIG_FILE_OPTIONS, NODE_REGION_CONFIG_OPTIONS } =
 require("@smithy/config-resolver");

const gremlin = require('gremlin');
const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;
const traversal = gremlin.process.AnonymousTraversalSource.traversal;
```

```
const DEFAULT_REGION = 'us-east-1';

const getCredentials = async () => {
    try {
        // Loads the default credential provider chain
        return await fromNodeProviderChain();
    } catch (e) {
        console.error("No credentials found", e);
        throw e;
    }
};

(main = async () => {
    console.log('Starting');

    const credentials = await getCredentials();
    try {
        // region set inside config profile or via AWS_REGION environment variable will
 be loaded
        credentials['region'] = await loadConfig(NODE_REGION_CONFIG_OPTIONS,
 NODE_REGION_CONFIG_FILE_OPTIONS)();
    } catch (e) {
        credentials['region'] = DEFAULT_REGION
    }

    const connInfo = getUrlAndHeaders(
        'you.cluster.endpoint.neptune.amazonaws.com',
        '8182',
        credentials,
        '/gremlin',
        'wss');

    const dc = new DriverRemoteConnection(connInfo['url'], {headers:
 connInfo['headers']});
    const g = traversal().withRemote(dc);

    const query = g.V().limit(5).count();
    const count = await query.next();
    console.log("Vertex count: " + count.value);

    await dc.close();
})
```

```
main();
```

Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Gremlin Go

Présentation de

Ce guide explique comment se connecter à une base de données Amazon Neptune avec l'authentification IAM activée à l'aide du pilote G705 Go, avec l'authentification Signature version 4 et le AWS SDK pour GO v2.

Conditions préalables

- Un cluster Amazon Neptune sur lequel l'authentification IAM est activée.
- Go 1.22 ou version ultérieure (reportez-vous aux versions minimales prises en charge pour Gremlin Go et AWS SDK for Go v2).
- AWS informations d'identification configurées (via des variables d'environnement, un fichier d'informations d'identification partagé ou un rôle IAM)

Création d'une connexion de base

Utilisez l'exemple de code suivant pour savoir comment établir une connexion de base avec l'authentification IAM à l'aide du pilote Gremlin Go.

```
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-sdk-go-v2/config"
 "net/http"
 "strings"
 "time"

 gremlingo "github.com/apache/tinkerpop/gremlin-go/v3/driver"
 v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
)

const emptyStringSHA256 =
 `e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855`
```

```
func main() {
 neptuneEndpoint := "you.cluster.endpoint.neptune.amazonaws.com"
 connString := "wss://" + neptuneEndpoint + ":8182/gremlin"
 service := "neptune-db"
 defaultRegion := "us-east-1"

 // Create request to sign
 req, err := http.NewRequest(http.MethodGet, connString, strings.NewReader(""))
 if err != nil {
  fmt.Println(err)
  return
 }

 // Loads the default config with default credentials provider
 // See https://github.com/aws/aws-sdk-go-v2 for additional docs on API usage
 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println(err)
  return
 }
 // Retrieve loaded credentials
 cr, err := cfg.Credentials.Retrieve(context.TODO())
 if err != nil {
  fmt.Println(err)
  return
 }

 region := defaultRegion
 if cfg.Region != "" {
  // region set inside config profile, or via AWS_REGION or AWS_DEFAULT_REGION
 environment variable will be loaded
  region = cfg.Region
 }

 signer := v4.NewSigner()
 // Sign request
 err = signer.SignHTTP(context.TODO(), cr, req, emptyStringSHA256, service, "us-
east-2", time.Now())
 if err != nil {
  fmt.Println(err)
  return
 }
```

```
 // Pass the signed request header into gremlingo.HeaderAuthInfo()
 driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection(connString,
  func(settings *gremlingo.DriverRemoteConnectionSettings) {
   settings.TraversalSource = "g"
   settings.AuthInfo = gremlingo.HeaderAuthInfo(req.Header)
   // settings.TlsConfig = &tls.Config{InsecureSkipVerify: true} // Use this only if
 you're on a Mac running Go 1.18+ doing local dev. See https://github.com/golang/go/
issues/51991
  })
 if err != nil {
  fmt.Println(err)
  return
 }

 // Cleanup
 defer driverRemoteConnection.Close()

 // Creating graph traversal
 g := gremlingo.Traversal_().WithRemote(driverRemoteConnection)

 // Query execution
 count, err := g.V().Limit(5).Count().Next()
 if err != nil {
  fmt.Println(err)
  return
 }
 fmt.Println("Vertex count:", *count)
}
```

Actualisation dynamique des informations d'identification G705 Go

G705 Go permet d'injecter un pointeur de fonction pour récupérer les informations d'identification et générer l'en-tête, ce qui empêche l'expiration de l'en-tête en cas de connexions de longue durée. DynamicAuth

```
package main

import (
 "context"
 "crypto/tls"
 "fmt"
 "github.com/aws/aws-sdk-go-v2/config"
 "net/http"
```

```
 "strings"
 "time"

 gremlingo "github.com/apache/tinkerpop/gremlin-go/v3/driver"
 v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
)

const emptyStringSHA256 =
 `e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855`

func main() {
 neptuneEndpoint := "you.cluster.endpoint.neptune.amazonaws.com"
 connString := "wss://" + neptuneEndpoint + ":8182/gremlin"
 service := "neptune-db"
 defaultRegion := "us-east-1"

 //Create the request to sign
 req, err := http.NewRequest(http.MethodGet, connString, strings.NewReader(""))
 if err != nil {
  fmt.Println(err)
  return
 }

 // Loads the default config with default credentials provider
 // See https://github.com/aws/aws-sdk-go-v2 for additional docs on API usage
 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println(err)
  return
 }

 region := defaultRegion
 if cfg.Region != "" {
  // region set inside config profile, or via AWS_REGION or AWS_DEFAULT_REGION
 environment variable will be loaded
  region = cfg.Region
 }

 signer := v4.NewSigner()

 // This is the function that will be used for dynamic refreseh of credentials and
 signed headers
 gen := func() gremlingo.AuthInfoProvider {
  // Retrieve loaded credentials
```

```
  cr, err := cfg.Credentials.Retrieve(context.TODO())
  fmt.Println("AWS Credentials: ", cr)
  if err != nil {
   fmt.Println(err)
   return
  }
  // Sign request
  err = signer.SignHTTP(context.TODO(), cr, req, emptyStringSHA256, service, region,
 time.Now())
  if err != nil {
   fmt.Println(err)
   return
  }
  fmt.Println(req.Header)
  return gremlingo.HeaderAuthInfo(req.Header)
 }

 // Pass the function into gremlingo.NewDynamicAuth(), which will generate the
 AuthInfoProvider to pass into gremlingo.DriverRemoteConnectionSettings below
 auth := gremlingo.NewDynamicAuth(gen)

 driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection(connString,
  func(settings *gremlingo.DriverRemoteConnectionSettings) {
   settings.TraversalSource = "g"
   settings.AuthInfo = auth
   // settings.TlsConfig = &tls.Config{InsecureSkipVerify: true} // Use this only if
 you're on a Mac running Go 1.18+ doing local dev. See https://github.com/golang/go/
issues/51991

  })
 if err != nil {
  fmt.Println(err)
  return
 }

 // Cleanup
 defer driverRemoteConnection.Close()

 // Creating graph traversal
 g := gremlingo.Traversal_().WithRemote(driverRemoteConnection)

 // Query execution
 count, err := g.V().Limit(5).Count().Next()
 if err != nil {
```

```
  fmt.Println(err)
  return
 }
 fmt.Println("Vertex count:", *count)
}
```

Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Gremlin .NET

Présentation de

Ce guide explique comment se connecter à une base de données Amazon Neptune avec l'authentification IAM activée à l'aide du pilote .NET G705, avec l'authentification Signature version 4 et le SDK pour AWS .NET v3.

Création d'une connexion de base

Pour vous connecter à Gremin.net, utilisez le fichier `SigV4RequestSigner` source de la bibliothèque https://github.com/aws/amazon-neptune-gremlin-dotnet-sigv4personnalisée. Un exemple de configuration de projet se trouve dans https://github.com/aws/amazon-neptune-gremlin-dotnet-sigv4/tree/main/example, et est reflété ci-dessous.

Dossier du projet :

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp8.0</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <Compile Include="../src/SigV4RequestSigner.cs" Link="SigV4RequestSigner.cs" />
    <PackageReference Include="AWSSDK.Core" Version="3.7.402.24" />
    <PackageReference Include="gremlin.net" Version="3.7.3" />
  </ItemGroup>

</Project>
```

Exemple de programme :

```
using System;
using System.Collections.Generic;
using System.Net.WebSockets;
using System.Linq;
using System.Net.Http;
using System.Security.Cryptography;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Web;
using Gremlin.Net;
using Gremlin.Net.Driver;
using Gremlin.Net.Driver.Remote;
using Gremlin.Net.Process;
using Gremlin.Net.Process.Traversal;
using Gremlin.Net.Structure;
using static Gremlin.Net.Process.Traversal.AnonymousTraversalSource;
using static Gremlin.Net.Process.Traversal.__;
using static Gremlin.Net.Process.Traversal.P;
using static Gremlin.Net.Process.Traversal.Order;
using static Gremlin.Net.Process.Traversal.Operator;
using static Gremlin.Net.Process.Traversal.Pop;
using static Gremlin.Net.Process.Traversal.Scope;
using static Gremlin.Net.Process.Traversal.TextP;
using static Gremlin.Net.Process.Traversal.Column;
using static Gremlin.Net.Process.Traversal.Direction;
using static Gremlin.Net.Process.Traversal.T;
using Amazon.Runtime.CredentialManagement;
using Amazon.Runtime;
using Amazon;
using Amazon.Util;
using Amazon.Neptune.Gremlin.Driver;

namespace NeptuneExample
{
    class Program
    {
        static void Main(string[] args)
        {
            /*
                Include your Neptune endpoint and port below.
            */
```

```
            var neptune_host = "neptune-endpoint"; // ex: mycluster.cluster.us-
east-1.neptune.amazonaws.com
            var neptune_port = 8182;

            var gremlinServer = new GremlinServer(neptune_host, neptune_port);
            var gremlinClient = new GremlinClient(gremlinServer,
 webSocketConfiguration: new SigV4RequestSigner().signRequest(neptune_host,
 neptune_port));
            var remoteConnection = new DriverRemoteConnection(gremlinClient);
            var g = Traversal().WithRemote(remoteConnection);

            /* Example code to pull the first 5 vertices in a graph. */
            Console.WriteLine("Get List of Node Labels:");
            Int32 limitValue = 5;
            var output = g.V().Limit<Vertex>(limitValue).ToList();
            foreach(var item in output) {
                Console.WriteLine(item);
            }
        }
    }
}
```

Gestion de l'accès aux bases de données Amazon Neptune à l'aide des politiques IAM

Les politiques IAM sont des objets JSON qui définissent les autorisations d'utilisation des actions et des ressources.

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique définit les autorisations lorsqu'elles sont associées à une identité ou à une ressource. AWS évalue ces politiques lorsqu'un directeur fait une demande. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations les documents de politique JSON, consultez Vue d'ensemble des politiques JSON dans le Guide de l'utilisateur IAM.

À l'aide de politiques, les administrateurs précisent qui a accès à quoi en définissant quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Un administrateur IAM crée des politiques IAM et les ajoute aux rôles, que les utilisateurs peuvent ensuite assumer. Les

politiques IAM définissent les autorisations quelle que soit la méthode que vous utilisez pour exécuter l'opération.

Identity-Based Politiques

Identity-based les politiques sont des documents de politique d'autorisation JSON que vous attachez à une identité (utilisateur, groupe ou rôle). Ces politiques contrôlent les actions que peuvent exécuter ces identités, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Définition d'autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l'utilisateur IAM.

Identity-based les politiques peuvent être des politiques intégrées (intégrées directement dans une seule identité) ou des politiques gérées (politiques autonomes associées à plusieurs identités). Pour découvrir comment choisir entre des politiques gérées et en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

Utilisation des politiques de contrôle des services (SCP) avec AWS organisations

Les politiques de contrôle des services (SCP) sont des politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée plusieurs AWS comptes détenus par votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer les politiques de contrôle des services (SCP) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités des comptes membres, y compris pour chaque utilisateur root AWS du compte. Pour plus d'informations sur les Organizations et les SCP, consultez How SCP work dans le Guide de l' AWS Organizations utilisateur.

Les clients déployant Amazon Neptune dans un AWS compte au sein d'une AWS organisation peuvent utiliser les SCP pour contrôler quels comptes peuvent utiliser Neptune. Pour garantir l'accès à Neptune depuis un compte de membre, assurez-vous de :

- Autoriser l'accès aux opérations `rds:*` de base de données Neptune et `neptune-db:*` pour celles-ci. Reportez-vous à la section Pourquoi les autorisations et les ressources Amazon RDS sont-elles nécessaires pour utiliser la base de données Neptune ? pour savoir pourquoi les autorisations Amazon RDS sont nécessaires pour la base de données Neptune.
- Autoriser l'accès `neptune-graph:*` aux opérations de Neptune Analytics.

Autorisations requises pour utiliser la console Amazon Neptune

Pour qu'un utilisateur puisse utiliser la console Amazon Neptune, il doit disposer d'un ensemble minimum d'autorisations. Ces autorisations lui permettront de décrire les ressources Neptune de son compte AWS et de fournir d'autres informations associées, y compris les informations relatives à la sécurité et au réseau Amazon EC2.

Si vous créez une politique IAM plus restrictive que les autorisations minimales requises, la console ne fonctionnera pas comme prévu pour les utilisateurs dotés de cette politique IAM. Pour que ces utilisateurs puissent continuer à utiliser la console Neptune, attachez également la politique gérée `NeptuneReadOnlyAccess` aux différents utilisateurs, comme indiqué dans la section Utilisation AWS politiques gérées pour accéder aux bases de données Amazon Neptune.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API Amazon Neptune AWS CLI ou l'API Amazon Neptune.

Association d'une politique IAM à un utilisateur IAM

Pour appliquer une politique personnalisée ou gérée, vous devez l'attacher à un utilisateur IAM. Pour accéder à un didacticiel sur ce sujet, consultez Créer et attacher votre première politique gérée par le client dans le Guide de l'utilisateur IAM.

Tandis que vous parcourez ce didacticiel, vous pouvez utiliser un exemple de politique illustré dans cette section comme point de départ afin de le personnaliser en fonction de vos besoins. À la fin de ce didacticiel, vous aurez un utilisateur IAM avec une politique associée qui peut utiliser l'action `neptune-db:*`.

> **Important**
>
> - Il faut jusqu'à 10 minutes pour que les modifications apportées à une politique IAM s'appliquent aux ressources Neptune spécifiées.
> - Les politiques IAM appliquées à un cluster de bases de données Neptune s'appliquent à toutes les instances de ce cluster.

Utilisation de différents types de politique IAM pour contrôler l'accès à Neptune

Pour donner accès aux actions administratives Neptune ou aux données d'un cluster de bases de données Neptune, vous devez associer des politiques à un utilisateur ou à un rôle IAM. Pour en savoir plus sur la façon d'associer une politique IAM à un utilisateur, consultez Association d'une politique IAM à un utilisateur IAM. Pour en savoir plus sur l'association d'une politique à un rôle, consultez Ajout et suppression de politiques IAM dans le Guide de l'utilisateur IAM.

Pour un accès général à Neptune, vous pouvez utiliser l'une des politiques gérées de Neptune. Pour un accès plus restreint, vous pouvez créer votre propre politique personnalisée à l'aide des actions administratives et des ressources administratives prises en charge par Neptune.

Dans une politique IAM personnalisée, vous pouvez utiliser deux types de déclarations de politique qui contrôlent différents modes d'accès à un cluster de bases de données Neptune :

- Déclarations de politique administrative : les déclarations de politique administrative donnent accès aux API de gestion Neptune que vous utilisez pour créer, configurer et gérer un cluster de bases de données et ses instances.

  Comme Neptune partage des fonctionnalités avec Amazon RDS, les actions administratives, les ressources et les clés de condition des politiques Neptune utilisent un préfixe `rds:` dès leur conception.
- Data-access déclarations de politique : les déclarations de Data-access politique utilisent des actions d'accès aux données, des ressources et des clés de condition pour contrôler l'accès aux données contenues dans un cluster de base de données.

  Les actions d'accès aux données, les ressources et les clés de condition Neptune utilisent un préfixe `neptune-db:`.

Utilisation des clés contextuelles de condition IAM dans Amazon Neptune

Vous pouvez spécifier des conditions dans une déclaration de politique IAM qui contrôle l'accès à Neptune. La déclaration de politique n'entre ensuite en vigueur que lorsque les conditions sont remplies.

Par exemple, il peut être utile d'appliquer une déclaration de politique après une date spécifique ou de n'autoriser l'accès que lorsqu'une valeur spécifique est présente dans la demande.

Pour exprimer des conditions, utilisez les clés de condition prédéfinies dans l'élément `Condition` d'une déclaration de politique avec des opérateurs de politique de condition IAM comme « égal à » et « inférieur à ».

Si vous spécifiez plusieurs éléments `Condition` dans une instruction, ou plusieurs clés dans un seul élément `Condition`, AWS les évalue à l'aide d'une opération `AND` logique. Si vous spécifiez plusieurs valeurs pour une seule clé de condition, AWS évalue la condition à l'aide d'une `OR` opération logique. Toutes les conditions doivent être remplies avant que les autorisations associées à l'instruction ne soient accordées.

Vous pouvez aussi utiliser des variables d'espace réservé quand vous spécifiez des conditions. Par exemple, vous pouvez accorder à un utilisateur IAM l'autorisation d'accéder à une ressource uniquement si elle est balisée avec son nom d'utilisateur IAM. Pour plus d'informations, consultez Éléments d'une politique IAM : variables et balises dans le Guide de l'utilisateur IAM.

Le type de données d'une clé de condition détermine les opérateurs de condition que vous pouvez utiliser pour comparer les valeurs de la demande avec les valeurs de la déclaration de politique. Si vous utilisez un opérateur de condition incompatible avec ce type de données, la correspondance échoue toujours et la déclaration de politique ne s'applique jamais.

Neptune prend en charge différents ensembles de clés de condition pour les déclarations de politique administrative et pour les déclarations de stratégie d'accès aux données :

- Clés de condition des déclarations de politique administrative
- Clés de condition des déclarations de stratégie d'accès aux données

Prise en charge des fonctionnalités de politique IAM et de contrôle d'accès dans Amazon Neptune

Le tableau suivant indique les fonctionnalités IAM prises en charge par Neptune pour les déclarations de politique administrative et les déclarations de stratégie d'accès aux données :

Fonctionnalités IAM que vous pouvez utiliser avec Neptune

Fonctionnalité IAM	Administration	Data-access
Identity-based politiques	Oui	Oui
Resource-based politiques	Non	Non

Fonctionnalité IAM	Administration	Data-access
Actions de politique	Oui	Oui
Ressources de politique	Oui	Oui
Clés de condition globale	Oui	(sous-ensemble)
Tag-based clés de condition	Oui	Non
Listes de contrôle d'accès (ACL)	Non	Non
Politiques de contrôle de service (SCP)	Oui	Oui
Rôles liés à un service	Oui	Non

Limites des politiques IAM

Il faut jusqu'à 10 minutes pour que les modifications apportées à une politique IAM s'appliquent aux ressources Neptune spécifiées.

Les politiques IAM appliquées à un cluster de bases de données Neptune s'appliquent à toutes les instances de ce cluster.

Neptune ne prend actuellement pas en charge le contrôle d'accès entre comptes au niveau du plan de données. Cross-account le contrôle d'accès n'est pris en charge que lors du chargement groupé et en utilisant le chaînage des rôles. Pour plus d'informations, consultez le didacticiel sur le chargement en bloc.

Utilisation AWS politiques gérées pour accéder aux bases de données Amazon Neptune

AWS répond à de nombreux cas d'utilisation courants en fournissant des politiques IAM autonomes créées et administrées par. AWS Les politiques gérées octroient les autorisations requises dans les cas d'utilisation courants et vous évitent d'avoir à réfléchir aux autorisations qui sont requises. Pour plus d'informations, consultez Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

Les politiques AWS gérées suivantes, que vous pouvez associer aux utilisateurs de votre compte, concernent l'utilisation des API de gestion Amazon Neptune :

- NeptuneReadOnlyAccess— Accorde des actions administratives en lecture seule (telles que `rds:Describe*` et`rds:ListTagsForResource`) et des actions d'accès aux données en lecture seule (`neptune-db:Read*``neptune-db:Get*`, et) `neptune-db:List*` sur toutes les ressources Neptune. Utilisez cette politique pour les utilisateurs qui doivent consulter les configurations de clusters et interroger des données sans apporter de modifications.
- NeptuneFullAccess— Autorise toutes les actions administratives (`rds:*`sur les ressources Neptune) et toutes les actions d'accès aux données (). `neptune-db:*` Cette politique convient aux administrateurs qui gèrent les clusters Neptune via le SDK AWS CLI ou qui n'ont pas besoin d'y accéder. AWS Management Console
- NeptuneConsoleFullAccess— Accorde toutes les actions administratives sur les ressources Neptune ainsi que les autorisations supplémentaires nécessaires aux flux de travail pour Amazon EC2 (VPC), IAM et Neptune Analytics. AWS Management Console Cette politique n'inclut pas les actions d'accès aux données (`neptune-db:*`). Utilisez cette politique pour les utilisateurs qui gèrent Neptune via la console de AWS gestion.
- NeptuneGraphReadOnlyAccess— Cette politique s'applique à Neptune Analytics. Pour plus de détails, voir NeptuneGraphReadOnlyAccessNeptune Analytics.
- AWSServiceRoleForNeptuneGraphPolicy— Cette politique s'applique à Neptune Analytics. Pour plus de détails, voir AWSServiceRoleForNeptuneGraphPolicyNeptune Analytics.

Les rôles et politiques IAM Neptune accordent un accès aux ressources Amazon RDS, car Neptune partage sa technologie opérationnelle avec Amazon RDS pour certaines fonctionnalités de gestion. Cela inclut les autorisations d'API administratives. C'est pourquoi les actions administratives Neptune ont le préfixe `rds:`.

Création de politiques personnalisées

Si les politiques AWS gérées sont trop larges pour votre cas d'utilisation, vous pouvez créer des politiques IAM personnalisées qui n'accordent que les autorisations spécifiques dont vous avez besoin. Neptune prend en charge deux catégories de politiques personnalisées :

- Politiques administratives : contrôlez l'accès aux opérations de gestion de Neptune, telles que la création, la modification et la suppression de clusters et d'instances. Ces actions utilisent le `rds:` préfixe. Pour obtenir des exemples, consultez the section called “Exemples de politiques administratives”.

- Data-access politiques — Contrôlez l'accès aux données de votre base de données de graphes Neptune, y compris les opérations de lecture, d'écriture et de suppression. Ces actions utilisent le `neptune-db:` préfixe. Pour obtenir des exemples, consultez the section called “Data-access exemples de politiques”.

En combinant les déclarations de politique d'administration et d'accès aux données, vous pouvez accorder des autorisations précises adaptées à chaque utilisateur ou rôle au sein de votre organisation.

Validation de politiques IAM

Lorsque vous créez ou modifiez des politiques IAM personnalisées, nous vous recommandons de les valider avant de les appliquer aux utilisateurs, aux groupes ou aux rôles.

Validation des politiques IAM Access Analyzer — IAM Access Analyzer fournit des vérifications des politiques qui valident vos politiques IAM par rapport à la grammaire des politiques IAM et aux meilleures pratiques. AWS Il identifie les erreurs, les avertissements de sécurité et les suggestions pour vous aider à créer des politiques fonctionnelles et conformes aux meilleures pratiques de sécurité. Pour plus d'informations, consultez la section Validation des politiques avec IAM Access Analyzer dans le guide de l'utilisateur d'IAM.

Simulateur de politique IAM — Le simulateur de politique IAM vous permet de tester les effets des politiques IAM avant de les mettre en production. Vous pouvez simuler des appels d'API à AWS des services pour vérifier que vos politiques accordent ou refusent l'accès attendu. Pour plus d'informations, consultez la section Tester les politiques IAM avec le simulateur de politique IAM dans le guide de l'utilisateur IAM.

Mises à jour de Neptune AWS stratégies gérées

Le tableau suivant effectue le suivi des mises à jour des politiques gérées de Neptune depuis le moment où Neptune a commencé à suivre ces modifications :

Politique	Description	Date
AWS politiques gérées pour Amazon Neptune - mise à jour des politiques existantes	Les politiques `NeptuneFullAccess` gérées `NeptuneReadOnlyAcess` et incluent désormais `Sid`	22/01/2022

Politique	Description	Date
	(ID de déclaration) comme identifiant dans la déclaration de politique.	
NeptuneGraphReadOnlyAccess(publié)	Publié pour fournir un accès en lecture seule aux graphiques et aux ressources de Neptune Analytics.	29/11/2023
AWSServiceRoleForNeptuneGraphPolicy(publié)	Publié pour permettre aux graphes de Neptune Analytics d'accéder à des statistiques et CloudWatch à des journaux opérationnels et d'utilisation. Consultez Utilisation de rôles liés à un service (SLR) dans Neptune Analytics.	29/11/2023
NeptuneConsoleFullAccess(autorisations ajoutées)	Les autorisations ajoutées fournissent tous les accès nécessaires pour interagir avec les graphes de Neptune Analytics.	2023- 11/29
NeptuneFullAccess(autorisations ajoutées)	Ajout d'autorisations d'accès aux données et d'autorisations pour les nouvelles API de base de données globales.	07-28
NeptuneConsoleFullAccess(autorisations ajoutées)	Autorisations ajoutées pour les nouvelles API de base de données globales.	07-21
Début du suivi des modifications avec Neptune	Neptune a commencé à suivre les modifications apportées à ses politiques AWS gérées.	07-21

Octroi **NeptuneReadOnlyAccess**à Amazon Neptune de bases de données à l'aide de AWS politique gérée

La politique NeptuneReadOnlyAccessgérée ci-dessous accorde un accès en lecture seule à toutes les actions et ressources de Neptune à des fins administratives et d'accès aux données.

> **Note**
>
> Cette politique a été mise à jour le 21 juillet pour inclure les autorisations d'accès aux données en lecture seule ainsi que les autorisations administratives en lecture seule et pour inclure les autorisations nécessaires pour les actions de base de données globales.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowReadOnlyPermissionsForRDS",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
```

```
                "rds:DownloadDBLogFilePortion",
                "rds:ListTagsForResource"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForCloudwatch",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForEC2",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForKMS",
            "Effect": "Allow",
            "Action": [
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "kms:ListAliases",
                "kms:ListKeyPolicies"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForLogs",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams",
```

```
                "logs:GetLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*",
                "arn:aws:logs:*:*:log-group:/aws/neptune/*:log-stream:*"
            ]
        },
        {
            "Sid": "AllowReadOnlyPermissionsForNeptuneDB",
            "Effect": "Allow",
            "Action": [
                "neptune-db:Read*",
                "neptune-db:Get*",
                "neptune-db:List*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Octroi **NeptuneFullAccess**à Amazon Neptune de bases de données à l'aide de AWS politique gérée

La politique NeptuneFullAccessgérée ci-dessous accorde un accès complet à toutes les actions et ressources de Neptune à des fins administratives et d'accès aux données. Il est recommandé si vous avez besoin d'un accès complet depuis AWS CLI ou depuis un SDK, mais pas depuis le AWS Management Console.

> **Note**
>
> Cette politique a été mise à jour le 21 juillet pour inclure des autorisations complètes d'accès aux données ainsi que des autorisations administratives complètes et pour inclure les autorisations nécessaires pour les actions de base de données globales.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowNeptuneCreate",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBCluster",
                "rds:CreateDBInstance"
            ],
            "Resource": [
                "arn:aws:rds:*:*:*"
            ],
            "Condition": {
                "StringEquals": {
                    "rds:DatabaseEngine": [
                        "graphdb",
                        "neptune"
                    ]
                }
            }
        },
        {
            "Sid": "AllowManagementPermissionsForRDS",
            "Effect": "Allow",
            "Action": [
                "rds:AddRoleToDBCluster",
                "rds:AddSourceIdentifierToSubscription",
                "rds:AddTagsToResource",
                "rds:ApplyPendingMaintenanceAction",
                "rds:CopyDBClusterParameterGroup",
                "rds:CopyDBClusterSnapshot",
                "rds:CopyDBParameterGroup",
                "rds:CreateDBClusterEndpoint",
                "rds:CreateDBClusterParameterGroup",
                "rds:CreateDBClusterSnapshot",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSubnetGroup",
                "rds:CreateEventSubscription",
                "rds:CreateGlobalCluster",
                "rds:DeleteDBCluster",
```

```
                "rds:DeleteDBClusterEndpoint",
                "rds:DeleteDBClusterParameterGroup",
                "rds:DeleteDBClusterSnapshot",
                "rds:DeleteDBInstance",
                "rds:DeleteDBParameterGroup",
                "rds:DeleteDBSubnetGroup",
                "rds:DeleteEventSubscription",
                "rds:DeleteGlobalCluster",
                "rds:DescribeDBClusterEndpoints",
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEngineDefaultClusterParameters",
                "rds:DescribeEngineDefaultParameters",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOptionGroups",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DescribeValidDBInstanceModifications",
                "rds:DownloadDBLogFilePortion",
                "rds:FailoverDBCluster",
                "rds:FailoverGlobalCluster",
                "rds:ListTagsForResource",
                "rds:ModifyDBCluster",
                "rds:ModifyDBClusterEndpoint",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyDBSubnetGroup",
```

```
                "rds:ModifyEventSubscription",
                "rds:ModifyGlobalCluster",
                "rds:PromoteReadReplicaDBCluster",
                "rds:RebootDBInstance",
                "rds:RemoveFromGlobalCluster",
                "rds:RemoveRoleFromDBCluster",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:RemoveTagsFromResource",
                "rds:ResetDBClusterParameterGroup",
                "rds:ResetDBParameterGroup",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:RestoreDBClusterToPointInTime",
                "rds:StartDBCluster",
                "rds:StopDBCluster"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "AllowOtherDepedentPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs",
                "kms:ListAliases",
                "kms:ListKeyPolicies",
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "sns:ListSubscriptions",
                "sns:ListTopics",
                "sns:Publish"
            ],
            "Resource": [
                "*"
            ]
```

```
        },
        {
            "Sid": "AllowPassRoleForNeptune",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:passedToService": "rds.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowCreateSLRForNeptune",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "rds.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowDataAccessForNeptune",
            "Effect": "Allow",
            "Action": [
                "neptune-db:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Octroi **NeptuneConsoleFullAccess**en utilisant AWS politique gérée

La politique NeptuneConsoleFullAccessgérée ci-dessous accorde un accès complet à toutes les actions et ressources de Neptune à des fins administratives, mais pas à des fins d'accès aux

données. Il inclut aussi des autorisations supplémentaires pour simplifier l'accès à Neptune depuis la console, y compris les autorisations IAM et EC2 (VPC) limitées.

> **Note**
>
> Cette politique a été mise à jour le 29-11-2023 afin d'inclure les autorisations nécessaires pour interagir avec les graphes de Neptune Analytics.
> Cette politique a été mise à jour le 21-07-2022 afin d'inclure des autorisations pour les actions de base de données globales.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowNeptuneCreate",
      "Effect": "Allow",
      "Action": [
        "rds:CreateDBCluster",
        "rds:CreateDBInstance"
      ],
      "Resource": [
        "arn:aws:rds:*:*:*"
      ],
      "Condition": {
        "StringEquals": {
          "rds:DatabaseEngine": [
            "graphdb",
            "neptune"
          ]
        }
      }
    },
    {
      "Sid": "AllowManagementPermissionsForRDS",
      "Action": [
        "rds:AddRoleToDBCluster",
        "rds:AddSourceIdentifierToSubscription",
        "rds:AddTagsToResource",
```

```
            "rds:ApplyPendingMaintenanceAction",
            "rds:CopyDBClusterParameterGroup",
            "rds:CopyDBClusterSnapshot",
            "rds:CopyDBParameterGroup",
            "rds:CreateDBClusterParameterGroup",
            "rds:CreateDBClusterSnapshot",
            "rds:CreateDBParameterGroup",
            "rds:CreateDBSubnetGroup",
            "rds:CreateEventSubscription",
            "rds:DeleteDBCluster",
            "rds:DeleteDBClusterParameterGroup",
            "rds:DeleteDBClusterSnapshot",
            "rds:DeleteDBInstance",
            "rds:DeleteDBParameterGroup",
            "rds:DeleteDBSubnetGroup",
            "rds:DeleteEventSubscription",
            "rds:DescribeAccountAttributes",
            "rds:DescribeCertificates",
            "rds:DescribeDBClusterParameterGroups",
            "rds:DescribeDBClusterParameters",
            "rds:DescribeDBClusterSnapshotAttributes",
            "rds:DescribeDBClusterSnapshots",
            "rds:DescribeDBClusters",
            "rds:DescribeDBEngineVersions",
            "rds:DescribeDBInstances",
            "rds:DescribeDBLogFiles",
            "rds:DescribeDBParameterGroups",
            "rds:DescribeDBParameters",
            "rds:DescribeDBSecurityGroups",
            "rds:DescribeDBSubnetGroups",
            "rds:DescribeEngineDefaultClusterParameters",
            "rds:DescribeEngineDefaultParameters",
            "rds:DescribeEventCategories",
            "rds:DescribeEventSubscriptions",
            "rds:DescribeEvents",
            "rds:DescribeOptionGroups",
            "rds:DescribeOrderableDBInstanceOptions",
            "rds:DescribePendingMaintenanceActions",
            "rds:DescribeValidDBInstanceModifications",
            "rds:DownloadDBLogFilePortion",
            "rds:FailoverDBCluster",
            "rds:ListTagsForResource",
            "rds:ModifyDBCluster",
            "rds:ModifyDBClusterParameterGroup",
```

```
        "rds:ModifyDBClusterSnapshotAttribute",
        "rds:ModifyDBInstance",
        "rds:ModifyDBParameterGroup",
        "rds:ModifyDBSubnetGroup",
        "rds:ModifyEventSubscription",
        "rds:PromoteReadReplicaDBCluster",
        "rds:RebootDBInstance",
        "rds:RemoveRoleFromDBCluster",
        "rds:RemoveSourceIdentifierFromSubscription",
        "rds:RemoveTagsFromResource",
        "rds:ResetDBClusterParameterGroup",
        "rds:ResetDBParameterGroup",
        "rds:RestoreDBClusterFromSnapshot",
        "rds:RestoreDBClusterToPointInTime"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowOtherDepedentPermissions",
      "Action": [
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:ListMetrics",
        "ec2:AllocateAddress",
        "ec2:AssignIpv6Addresses",
        "ec2:AssignPrivateIpAddresses",
        "ec2:AssociateAddress",
        "ec2:AssociateRouteTable",
        "ec2:AssociateSubnetCidrBlock",
        "ec2:AssociateVpcCidrBlock",
        "ec2:AttachInternetGateway",
        "ec2:AttachNetworkInterface",
        "ec2:CreateCustomerGateway",
        "ec2:CreateDefaultSubnet",
        "ec2:CreateDefaultVpc",
        "ec2:CreateInternetGateway",
        "ec2:CreateNatGateway",
        "ec2:CreateNetworkInterface",
        "ec2:CreateRoute",
        "ec2:CreateRouteTable",
        "ec2:CreateSecurityGroup",
        "ec2:CreateSubnet",
```

```
        "ec2:CreateVpc",
        "ec2:CreateVpcEndpoint",
        "ec2:CreateVpcEndpoint",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAddresses",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeCustomerGateways",
        "ec2:DescribeInstances",
        "ec2:DescribeNatGateways",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribePrefixLists",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroupReferences",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs",
        "ec2:DescribeVpcs",
        "ec2:ModifyNetworkInterfaceAttribute",
        "ec2:ModifySubnetAttribute",
        "ec2:ModifyVpcAttribute",
        "ec2:ModifyVpcEndpoint",
        "iam:ListRoles",
        "kms:ListAliases",
        "kms:ListKeyPolicies",
        "kms:ListKeys",
        "kms:ListRetirableGrants",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents",
        "sns:ListSubscriptions",
        "sns:ListTopics",
        "sns:Publish"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
```

```
    {
      "Sid": "AllowPassRoleForNeptune",
      "Action": "iam:PassRole",
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:passedToService": "rds.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowCreateSLRForNeptune",
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "rds.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowManagementPermissionsForNeptuneAnalytics",
      "Effect": "Allow",
      "Action": [
        "neptune-graph:CreateGraph",
        "neptune-graph:DeleteGraph",
        "neptune-graph:GetGraph",
        "neptune-graph:ListGraphs",
        "neptune-graph:UpdateGraph",
        "neptune-graph:ResetGraph",
        "neptune-graph:CreateGraphSnapshot",
        "neptune-graph:DeleteGraphSnapshot",
        "neptune-graph:GetGraphSnapshot",
        "neptune-graph:ListGraphSnapshots",
        "neptune-graph:RestoreGraphFromSnapshot",
        "neptune-graph:CreatePrivateGraphEndpoint",
        "neptune-graph:GetPrivateGraphEndpoint",
        "neptune-graph:ListPrivateGraphEndpoints",
        "neptune-graph:DeletePrivateGraphEndpoint",
        "neptune-graph:CreateGraphUsingImportTask",
        "neptune-graph:GetImportTask",
```

```
        "neptune-graph:ListImportTasks",
        "neptune-graph:CancelImportTask"
      ],
      "Resource": [
        "arn:aws:neptune-graph:*:*:*"
      ]
    },
    {
      "Sid": "AllowPassRoleForNeptuneAnalytics",
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:passedToService": "neptune-graph.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowCreateSLRForNeptuneAnalytics",
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::*:role/aws-service-role/neptune-
graph.amazonaws.com/AWSServiceRoleForNeptuneGraph",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "neptune-graph.amazonaws.com"
        }
      }
    }
  ]
}
```

Personnalisation de l'accès aux ressources Amazon Neptune à l'aide des clés contextuelles de condition IAM

Vous pouvez spécifier dans les politiques IAM des conditions qui contrôlent l'accès aux actions et ressources de gestion Neptune. La déclaration de politique n'entre ensuite en vigueur que lorsque les conditions sont remplies.

Par exemple, il peut être utile d'appliquer une déclaration de politique après une date spécifique ou de n'autoriser l'accès que lorsqu'une valeur spécifique est présente dans la demande d'API.

Pour exprimer des conditions, utilisez les clés de condition prédéfinies dans l'élément `Condition` d'une déclaration de politique avec des opérateurs de politique de condition IAM comme « égal à » et « inférieur à ».

Si vous spécifiez plusieurs éléments `Condition` dans une instruction, ou plusieurs clés dans un seul élément `Condition`, AWS les évalue à l'aide d'une opération `AND` logique. Si vous spécifiez plusieurs valeurs pour une seule clé de condition, AWS évalue la condition à l'aide d'une `OR` opération logique. Toutes les conditions doivent être remplies avant que les autorisations associées à l'instruction ne soient accordées.

Vous pouvez aussi utiliser des variables d'espace réservé quand vous spécifiez des conditions. Par exemple, vous pouvez accorder à un utilisateur IAM l'autorisation d'accéder à une ressource uniquement si elle est balisée avec son nom d'utilisateur IAM. Pour plus d'informations, consultez Éléments d'une politique IAM : variables et balises dans le Guide de l'utilisateur IAM.

Le type de données d'une clé de condition détermine les opérateurs de condition que vous pouvez utiliser pour comparer les valeurs de la demande avec les valeurs de la déclaration de politique. Si vous utilisez un opérateur de condition incompatible avec ce type de données, la correspondance échoue toujours et la déclaration de politique ne s'applique jamais.

Clés de condition IAM pour les déclarations de politique administrative Neptune

- Clés de condition globales : vous pouvez utiliser la plupart des clés de condition AWS globales dans les déclarations de politique administrative de Neptune.
- Service-specific clés de condition : il s'agit de clés définies pour des AWS services spécifiques. Celles que Neptune prend en charge pour les déclarations de politique administrative sont répertoriées dans Clés de condition IAM pour administrer Amazon Neptune.

Clés de condition IAM pour les déclarations de stratégie d'accès aux données Neptune

- Clés de condition globales : le sous-ensemble de ces clés pris en charge par Neptune dans les déclarations de stratégie d'accès aux données est répertorié dans AWS clés contextuelles de conditions globales prises en charge par Neptune dans les déclarations de politique d'accès aux données.
- Service-specific les clés de condition définies par Neptune pour les déclarations de politique d'accès aux données sont répertoriées dans. Clés de condition

Création de déclarations de politique IAM personnalisées pour administrer Amazon Neptune

Les déclarations de politique administrative vous permettent de contrôler ce qu'un utilisateur IAM peut faire pour gérer une base de données Neptune.

Une déclaration de politique administrative Neptune donne accès à une ou plusieurs actions administratives et ressources administratives prises en charge par Neptune. Vous pouvez également utiliser des Clés de condition pour rendre les autorisations administratives plus spécifiques.

> **Note**
>
> Comme Neptune partage des fonctionnalités avec Amazon RDS, les actions administratives, les ressources et les clés de condition spécifiques aux services figurant dans les déclarations de politique administrative utilisent un préfixe `rds:` dès leur conception.

Rubriques

- Actions IAM pour administrer Amazon Neptune
- Types de ressources IAM pour administrer Amazon Neptune
- Clés de condition IAM pour administrer Amazon Neptune
- Création de déclarations de politique administrative IAM pour Amazon Neptune

Actions IAM pour administrer Amazon Neptune

Vous pouvez utiliser les actions administratives répertoriées ci-dessous dans l'élément `Action`d'une déclaration de politique IAM pour contrôler l'accès aux API de gestion Neptune. Lorsque vous utilisez une action dans une politique, vous autorisez ou refusez généralement l'accès à l'opération d'API ou à la commande CLI portant le même nom. Toutefois, dans certains cas, une seule action contrôle l'accès à plusieurs opérations. D'autres opérations, quant à elles, requièrent plusieurs actions différentes.

Le champ `Resource type` de la liste ci-dessous indique si chaque action prend en charge les autorisations au niveau des ressources. S'il n'y a pas de valeur pour ce champ, vous devez indiquer toutes les ressources (*) dans l'élément `Resource` de la déclaration de politique. Si la colonne inclut un type de ressource, vous pouvez indiquer un ARN de ce type dans une déclaration avec cette action. Les types de ressources administratives Neptune sont répertoriés sur cette page.

Les ressources requises sont indiquées dans la liste ci-dessous par un astérisque (*). Si vous indiquez un ARN d'autorisation au niveau des ressources dans une déclaration à l'aide de cette action, il doit être de ce type. Certaines actions prennent en charge plusieurs types de ressources. Si un type de ressource est facultatif (en d'autres termes, s'il n'est pas marqué d'un astérisque), vous n'êtes pas obligé de l'inclure.

Pour plus d'informations sur les champs répertoriés ici, consultez le tableau des actions du Guide de l'utilisateur IAM.

rouge : AddRoleToDBCluster

`AddRoleToDBCluster` associe un rôle IAM à un cluster de bases de données Neptune.

Niveau d'accès : `Write`.

Actions dépendantes : `iam:PassRole`.

Type de ressource : cluster (obligatoire).

rouge : AddSourceIdentifierToSubscription

`AddSourceIdentifierToSubscription` ajoute un identifiant source à un abonnement existant à la notification d'événements.

Niveau d'accès : `Write`.

Type de ressource : es (obligatoire).

rouge : AddTagsToResource

`AddTagsToResource` associe un rôle IAM à un cluster de bases de données Neptune.

Niveau d'accès : `Write`.

Types de ressources :

- db
- es
- pg
- cluster-snapshot
- subgrp

Clés de condition :

- lois :RequestTag/*tag-key*
- lois : TagKeys

rouge : ApplyPendingMaintenanceAction

`ApplyPendingMaintenanceAction` applique une action de maintenance en attente à une ressource.

Niveau d'accès : `Write`.

Type de ressource : db (obligatoire).

rouge : CopyDBClusterParameterGroup

`Copier DBCluster ParameterGroup` copie le groupe de paramètres de cluster de bases de données spécifié.

Niveau d'accès : `Write`.

Type de ressource : cluster-pg (obligatoire).

rouge : CopyDBClusterSnapshot

`Copier DBCluster un instantané` copie un instantané d'un cluster de bases de données.

Niveau d'accès : `Write`.

Type de ressource : cluster-snapshot (obligatoire).

rouge : CopyDBParameterGroup

`DBParameterGroupe de copies` copie le groupe de paramètres de base de données spécifié.

Niveau d'accès : `Write`.

Type de ressource : pg (obligatoire).

rds:CreateDBCluster

`Créez DBCluster` crée un cluster de bases de données Amazon Neptune.

Niveau d'accès : `Tagging`.

Actions dépendantes : `iam:PassRole`.

Types de ressources :

- cluster (obligatoire).
- cluster-pg (obligatoire).
- subgrp (obligatoire).

Clés de condition :

- lois :RequestTag/*tag-key*
- lois : TagKeys
- neptune-rds_ DatabaseEngine

rouge : CreateDBClusterParameterGroup

`Créez DBCluster ParameterGroup` crée un groupe de paramètres de cluster de bases de données.

Niveau d'accès : `Tagging`.

Type de ressource : cluster-pg (obligatoire).

Clés de condition :

- lois :RequestTag/*tag-key*
- lois : TagKeys

rouge : CreateDBClusterSnapshot

`Créer un DBCluster instantané` crée un instantané d'un cluster de bases de données.

Niveau d'accès : `Tagging`.

Types de ressources :

- cluster (obligatoire).

- cluster-snapshot (obligatoire).

Clés de condition :

- lois :RequestTag/*tag-key*
- lois : TagKeys

rds:CreateDBInstance

`Créez DBInstance` crée une instance de base de données.

Niveau d'accès : `Tagging`.

Actions dépendantes : `iam:PassRole`.

Types de ressources :

- db (obligatoire).
- pg (obligatoire).
- subgrp (obligatoire).

Clés de condition :

- lois :RequestTag/*tag-key*
- lois : TagKeys

rouge : CreateDBParameterGroup

`Créer un DBParameter groupe` crée un groupe de paramètres de base de données.

Niveau d'accès : `Tagging`.

Type de ressource : pg (obligatoire).

Clés de condition :

- lois :RequestTag/*tag-key*
- lois : TagKeys

rouge : CreateDBSubnetGroup

`Créer un DBSubnet groupe` crée un groupe de sous-réseaux de base de données.

Niveau d'accès : `Tagging`.

Type de ressource : subgrp (obligatoire).

Clés de condition :

- lois :RequestTag/*tag-key*
- lois : TagKeys

rouge : CreateEventSubscription

`CreateEventSubscription` crée un abonnement aux notifications d'événement Neptune.

Niveau d'accès : `Tagging`.

Type de ressource : es (obligatoire).

Clés de condition :

- lois :RequestTag/*tag-key*
- lois : TagKeys

rds:DeleteDBCluster

`Supprimer DBCluster` supprime un cluster de bases de données Neptune existant.

Niveau d'accès : `Write`.

Types de ressources :

- cluster (obligatoire).
- cluster-snapshot (obligatoire).

rouge : DeleteDBClusterParameterGroup

`Supprimer DBCluster ParameterGroup` supprime un groupe de paramètres de cluster de bases de données spécifié.

Niveau d'accès : `Write`.

Type de ressource : cluster-pg (obligatoire).

rouge : DeleteDBClusterSnapshot

`Supprimer DBCluster un instantané` supprime un instantané de cluster de bases de données.

Niveau d'accès : `Write`.

Type de ressource : cluster-snapshot (obligatoire).

rds:DeleteDBInstance

`Supprimer DBInstance` supprime une instance de base de données spécifiée.

Niveau d'accès : `Write`.

Type de ressource : db (obligatoire).

rouge : DeleteDBParameterGroup

`Supprimer le DBParameter groupe`supprime un élément spécifié DBParameterGroup.

Niveau d'accès : `Write`.

Type de ressource : pg (obligatoire).

rouge : DeleteDBSubnetGroup

`Supprimer le DBSubnet groupe` supprime un groupe de sous-réseaux de base de données.

Niveau d'accès : `Write`.

Type de ressource : subgrp (obligatoire).

rouge : DeleteEventSubscription

`DeleteEventSubscription` supprime un abonnement aux notifications d'événements.

Niveau d'accès : `Write`.

Type de ressource : es (obligatoire).

rouge : DescribeDBClusterParameterGroups

Décrivez DBCluster ParameterGroupsrenvoie une liste de DBClusterParameterGroup descriptions.

Niveau d'accès : `List`.

Type de ressource : cluster-pg (obligatoire).

rouge : DescribeDBClusterParameters

Décrire DBCluster les paramètres renvoie la liste détaillée des paramètres d'un groupe de paramètres de cluster de bases de données spécifique.

Niveau d'accès : `List`.

Type de ressource : cluster-pg (obligatoire).

rouge : DescribeDBClusterSnapshotAttributes

Décrivez DBCluster SnapshotAttributes renvoie une liste de noms et de valeurs d'attributs d'instantané pour un instantané de cluster de bases de données manuel.

Niveau d'accès : `List`.

Type de ressource : cluster-snapshot (obligatoire).

rouge : DescribeDBClusterSnapshots

Décrire les DBCluster instantanés renvoie des informations sur des instantanés de cluster de bases de données.

Niveau d'accès : `Read`.

rds:DescribeDBClusters

Décrivez DBClusters renvoie des informations sur un cluster de bases de données Neptune provisionné.

Niveau d'accès : `List`.

Type de ressource : cluster (obligatoire).

rouge : DescribeDBEngineVersions

`Décrire DBEngine les versions` renvoie une liste des moteurs de base de données disponibles.

Niveau d'accès : `List`.

Type de ressource : pg (obligatoire).

rds:DescribeDBInstances

`Décrivez DBInstances` renvoie des informations sur les instances de base de données.

Niveau d'accès : `List`.

Type de ressource : es (obligatoire).

rouge : DescribeDBParameterGroups

`Décrire DBParameter les groupes`renvoie une liste de DBParameterGroup descriptions.

Niveau d'accès : `List`.

Type de ressource : pg (obligatoire).

rds:DescribeDBParameters

`Décrivez DBParameters` renvoie la liste détaillée des paramètres d'un groupe de paramètres de base de données spécifique.

Niveau d'accès : `List`.

Type de ressource : pg (obligatoire).

rouge : DescribeDBSubnetGroups

`Décrire DBSubnet les groupes`renvoie une liste de DBSubnetGroup descriptions.

Niveau d'accès : `List`.

Type de ressource : subgrp (obligatoire).

rouge : DescribeEventCategories

`DescribeEventCategories` affiche une liste des catégories de tous les types de sources d'événement ou, si la valeur est spécifiée, d'un type de source donné.

Niveau d'accès : `List`.

rouge : DescribeEventSubscriptions

`DescribeEventSubscriptions` répertorie toutes les descriptions d'abonnements d'un compte client.

Niveau d'accès : `List`.

Type de ressource : es (obligatoire).

rouge : DescribeEvents

`DescribeEvents` renvoie des événements associés aux instances de base de données, aux groupes de sécurité de base de données et aux groupes de paramètres de base de données des 14 derniers jours.

Niveau d'accès : `List`.

Type de ressource : es (obligatoire).

rouge : DescribeOrderableDBInstanceOptions

`DescribeOrderableDBInstanceOptions` renvoie une liste des options d'instance de base de données à commander pour le moteur spécifié.

Niveau d'accès : `List`.

rouge : DescribePendingMaintenanceActions

`DescribePendingMaintenanceActions` renvoie une liste des ressources (par exemple, des instances de base de données) ayant au moins une action de maintenance en attente.

Niveau d'accès : `List`.

Type de ressource : db (obligatoire).

rouge : DescribeValidDBInstanceModifications

`DescribeValidDBInstanceModifications` répertorie les modifications disponibles que vous pouvez apporter à votre instance de base de données.

Niveau d'accès : `List`.

Type de ressource : db (obligatoire).

rds:FailoverDBCluster

`Basculement DBCluster` force un basculement pour un cluster de bases de données.

Niveau d'accès : `Write`.

Type de ressource : cluster (obligatoire).

rouge : ListTagsForResource

`ListTagsForResource` répertorie toutes les balises sur une ressource Neptune.

Niveau d'accès : `Read`.

Types de ressources :

- cluster-snapshot
- db
- es
- pg
- subgrp

rds:ModifyDBCluster

`Modifier DBCluster`

Modifie un paramètre pour un cluster de bases de données Neptune.

Niveau d'accès : `Write`.

Actions dépendantes : `iam:PassRole`.

Types de ressources :

- cluster (obligatoire).
- cluster-pg (obligatoire).

rouge : ModifyDBClusterParameterGroup

`Modifier DBCluster ParameterGroup` modifie les paramètres d'un groupe de paramètres de cluster de bases de données.

Niveau d'accès : `Write`.

Type de ressource : cluster-pg (obligatoire).

rouge : ModifyDBClusterSnapshotAttribute

`Modifier DBCluster SnapshotAttribute` ajoute ou supprime un attribut et des valeurs dans un instantané de cluster de bases de données manuel.

Niveau d'accès : `Write`.

Type de ressource : cluster-snapshot (obligatoire).

rds:ModifyDBInstance

`Modifier DBInstance` modifie les paramètres d'une instance de base de données.

Niveau d'accès : `Write`.

Actions dépendantes : `iam:PassRole`.

Types de ressources :

- db (obligatoire).
- pg (obligatoire).

rouge : ModifyDBParameterGroup

`Modifier le DBParameter groupe` modifie les paramètres d'un groupe de paramètres de base de données.

Niveau d'accès : `Write`.

Type de ressource : pg (obligatoire).

rouge : ModifyDBSubnetGroup

`Modifier le DBSubnet groupe` modifie un groupe de sous-réseaux de base de données existant.

Niveau d'accès : `Write`.

Type de ressource : subgrp (obligatoire).

rouge : ModifyEventSubscription

ModifyEventSubscription modifie un abonnement aux notifications d'événements.

Niveau d'accès : Write.

Type de ressource : es (obligatoire).

rds:RebootDBInstance

Redémarrer DBInstance redémarre le service du moteur de base de données de l'instance.

Niveau d'accès : Write.

Type de ressource : db (obligatoire).

rouge : RemoveRoleFromDBCluster

RemoveRoleFromDBClusterdissocie un rôle AWS Identity and Access Management (IAM) d'un cluster de base de données Amazon Neptune.

Niveau d'accès : Write.

Actions dépendantes : iam:PassRole.

Type de ressource : cluster (obligatoire).

rouge : RemoveSourceIdentifierFromSubscription

RemoveSourceIdentifierFromSubscription supprime un identifiant source d'un abonnement aux notifications d'événements.

Niveau d'accès : Write.

Type de ressource : es (obligatoire).

rouge : RemoveTagsFromResource

RemoveTagsFromResource supprime des balises de métadonnées d'une ressource Neptune.

Niveau d'accès : Tagging.

Types de ressources :

- cluster-snapshot

- db
- es
- pg
- subgrp

Clés de condition :

- lois :RequestTag/*tag-key*
- lois : TagKeys

rouge : ResetDBClusterParameterGroup

`Réinitialiser DBCluster ParameterGroup` remplace les paramètres d'un groupe de paramètres de cluster de bases de données par les valeurs par défaut.

Niveau d'accès : `Write`.

Type de ressource : cluster-pg (obligatoire).

rouge : ResetDBParameterGroup

`Réinitialiser le DBParameter groupe`modifie les paramètres d'un groupe de paramètres de base de données à la valeur engine/system par défaut.

Niveau d'accès : `Write`.

Type de ressource : pg (obligatoire).

rouge : RestoreDBClusterFromSnapshot

`Restaurer DBCluster FromSnapshot` crée un cluster de bases de données à partir d'un instantané de cluster de bases de données.

Niveau d'accès : `Write`.

Actions dépendantes : `iam:PassRole`.

Types de ressources :

- cluster (obligatoire).
- cluster-snapshot (obligatoire).

Clés de condition :

- lois :RequestTag/*tag-key*
- lois : TagKeys

rouge : RestoreDBClusterToPointInTime

`Restaurer DBCluster ToPointInTime` restaure un cluster de bases de données à un moment donné arbitraire.

Niveau d'accès : `Write`.

Actions dépendantes : `iam:PassRole`.

Types de ressources :

- cluster (obligatoire).
- subgrp (obligatoire).

Clés de condition :

- lois :RequestTag/*tag-key*
- lois : TagKeys

rds:StartDBCluster

`Démarrer DBCluster` démarre le cluster de bases de données spécifié.

Niveau d'accès : `Write`.

Type de ressource : cluster (obligatoire).

rds:StopDBCluster

`Arrêtez DBCluster` arrête le cluster de bases de données spécifié.

Niveau d'accès : `Write`.

Type de ressource : cluster (obligatoire).

Types de ressources IAM pour administrer Amazon Neptune

Neptune prend en charge les types de ressources du tableau suivant. Ils peuvent être utilisés dans l'élément `Resource` des déclarations de politique d'administration IAM. Pour plus d'informations sur l'élément `Resource`, consultez Éléments de politique JSON IAM : Ressource.

La liste des actions d'administration Neptune identifie les types de ressources pouvant être spécifiés avec chaque action. Un type de ressource détermine également les clés de condition que vous pouvez inclure dans une politique, comme indiqué dans la dernière colonne du tableau suivant.

La colonne ARN dans le tableau ci-dessous spécifie le format Amazon Resource Name (ARN) à utiliser pour référencer des ressources de ce type. Les parties précédées d'un symbole `$` doivent être remplacées par les valeurs réelles de votre scénario. Par exemple, si `$user-name` s'affiche dans un ARN, vous devez remplacer cette chaîne par le nom de l'utilisateur IAM ou par une variable de politique contenant le nom d'un utilisateur IAM. Pour plus d'informations sur les ARN, consultez ARN IAM et Utilisation des ARN administratifs dans Amazon Neptune.

La colonne `Condition Keys` spécifie les clés de contexte de condition que vous pouvez inclure dans une déclaration de politique IAM uniquement lorsque cette action et cette ressource sont incluses dans la déclaration.

Types de ressources	ARN	Clés de condition
`cluster` (Cluster de bases de données)	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :cluster: `*`instance-name`*	lois :Resource Tag/*`tag-key`* rds : cluster-tag/ *`tag-key`*
`cluster-pg` (Groupe de paramètres de cluster de bases de données)	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :cluster-pg: `*`neptune-DBClusterParameterGroupName`*	lois :Resource Tag/*`tag-key`*

Types de ressources	ARN	Clés de condition
cluster-snapshot (Instantané de cluster de bases de données)	arn:*partition* :rds:*region*:*account-id* :cluster-snapshot: *neptune-DBClusterSnapshotName*	lois :ResourceTag/*tag-key* rds:cluster-snapshot-tag/ *tag-key*
db (instance de base de données)	arn:*partition* :rds:*region*:*account-id* :db:*neptune-DbInstanceName*	lois :ResourceTag/*tag-key* rouge : DatabaseClass rouge : DatabaseEngine rds : db-tag/ *tag-key*
es (abonnement aux événements)	arn:*partition* :rds:*region*:*account-id* :es:*neptune-CustSubscriptionId*	lois :ResourceTag/*tag-key* rds : es-tag/ *tag-key*
pg (Groupe de paramètres de base de données)	arn:*partition* :rds:*region*:*account-id* :pg:*neptune-ParameterGroupName*	lois :ResourceTag/*tag-key* rds : pg-tag/ *tag-key*

Types de ressources	ARN	Clés de condition
subgrp (Groupe de sous-réseaux de base de données)	arn:*partition* :rds:*region*:*account-id* :subgrp:*neptune-DBSubnetGroupName* }	lois :Resource Tag/*tag-key* rds : subgrp-tag/ *tag-key*

Clés de condition IAM pour administrer Amazon Neptune

À l'aide des clés de condition, vous pouvez spécifier des conditions dans une déclaration de politique IAM de sorte qu'elle ne prenne effet que lorsque les conditions sont remplies. Les clés de condition que vous pouvez utiliser dans les déclarations de politique administrative Neptune appartiennent aux catégories suivantes :

- Clés de condition globales : elles sont définies AWS pour une utilisation générale avec AWS les services. La plupart peuvent être utilisées dans les déclarations de politique administrative Neptune.
- Clés de condition des propriétés des ressources administratives : ces clés, répertoriées ci-dessous, sont basées sur les propriétés des ressources administratives.
- Tag-based clés de condition d'accès : ces clés, répertoriées ci-dessous, sont basées sur des AWS balises associées aux ressources administratives.

Clés de condition des propriétés des ressources administratives Neptune

Clés de condition	Description	Type
rds:DatabaseClass	Filtre l'accès en fonction du type de classe d'instances de base de données.	String
rds:DatabaseEngine	Filtre l'accès en fonction du moteur de base de données. Pour les valeurs possibles, reportez-vous au paramètre moteur dans l'API CreateDBInstance	Chaîne

Clés de condition	Description	Type
`rds:DatabaseName`	Filtre l'accès en fonction du nom défini par l'utilisateur de la base de données sur l'instance de base de données	Chaîne
`rds:EndpointType`	Filtre l'accès en fonction du type du point de terminaison. Exemple : LECTEUR, AUTEUR ou PERSONNALISÉ.	String
`rds:Vpc`	Filtre l'accès en fonction de la valeur qui spécifie si l'instance de base de données s'exécute dans un VPC Amazon (Amazon Virtual Private Cloud). Pour indiquer que l'instance de base de données s'exécute dans un VPC Amazon, spécifiez `true`.	Booléen

Clés de condition basées sur des balises administratives

Amazon Neptune prend en charge la spécification de conditions dans une politique IAM à l'aide de balises personnalisées, afin de contrôler l'accès à Neptune via l'Référence de l'API de gestion.

Par exemple, si vous ajoutez une balise nommée `environment` à vos instances de base de données, avec des valeurs telles que `beta`, `staging` et `production`, vous pouvez créer une politique qui restreint l'accès aux instances en fonction de la valeur de cette balise.

> **Important**
>
> Si vous gérez l'accès à vos ressources Neptune à l'aide du balisage, assurez-vous de sécuriser l'accès aux balises. Vous pouvez gérer l'accès aux balises en créant des politiques pour les actions `AddTagsToResource` et `RemoveTagsFromResource`.
> Par exemple, vous pouvez utiliser la politique suivante pour empêcher les utilisateurs la capacité d'ajouter ou de supprimer des balises pour toutes les ressources. Vous pouvez ensuite créer des politiques pour autoriser des utilisateurs spécifiques à ajouter ou supprimer des balises.
>
> JSON

```
{ "Version":"2012-10-17",
  "Statement":[
    { "Sid": "DenyTagUpdates",
```

```
      "Effect": "Deny",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource":"*"
    }
  ]
}
```

Les clés de condition suivantes basées sur des balises ne fonctionnent qu'avec les ressources administratives figurant dans les déclarations de politique administrative.

Tag-based clés de condition administrative

Clés de condition	Description	Type
aws:RequestTag/${TagKey}	Filtre l'accès en fonction de la présence de paires clé-valeur de balise dans la demande.	String
aws:ResourceTag/${TagKey}	Filtre l'accès en fonction des paires clé-valeur de balise attachées à la ressource.	String
aws:TagKeys	Filtre l'accès en fonction de la présence de clés de balise dans la demande.	String
rds:cluster-pg-tag/${TagKey}	Filtre l'accès en fonction de la balise attachée à un groupe de paramètres de cluster de bases de données.	String
rds:cluster-snapshot-tag/${TagKey}	Filtre l'accès en fonction de la balise attachée à un instantané de cluster de bases de données.	String

Clés de condition	Description	Type
`rds:cluster-tag/${TagKey}`	Filtre l'accès en fonction de la balise attachée à un cluster de bases de données.	String
`rds:db-tag/${TagKey}`	Filtre l'accès en fonction de la balise attachée à une instance de base de données.	String
`rds:es-tag/${TagKey}`	Filtre l'accès en fonction de la balise attachée à un abonnement à un événement.	String
`rds:pg-tag/${TagKey}`	Filtre l'accès en fonction de la balise attachée à un groupe de paramètres de base de données.	String
`rds:req-tag/${TagKey}`	Filtre l'accès en fonction de l'ensemble de clés et de valeurs de balise pouvant être utilisées pour baliser une ressource.	String
`rds:secgrp-tag/${TagKey}`	Filtre l'accès en fonction de la balise attachée à un groupe de sécurité de base de données.	String
`rds:snapshot-tag/${TagKey}`	Filtre l'accès en fonction de la balise attachée à un instantané de base de données.	String
`rds:subgrp-tag/${TagKey}`	Filtre l'accès en fonction de la balise attachée à un groupe de sous-réseaux de base de données	String

Création de déclarations de politique administrative IAM pour Amazon Neptune

Exemples de politiques administratives générales

Les exemples suivants montrent comment créer des politiques administratives Neptune qui accordent des autorisations pour effectuer diverses actions de gestion sur un cluster de bases de données.

Politique empêchant un utilisateur IAM de supprimer une instance de base de données spécifiée

Voici un exemple de politique qui empêche un utilisateur IAM de supprimer une instance de base de données Neptune spécifiée :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyDeleteOneInstance",
      "Effect": "Deny",
      "Action": "rds:DeleteDBInstance",
      "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-instance-name"
    }
  ]
}
```

Politique autorisant la création d'instances de base de données

Voici un exemple de politique autorisant un utilisateur IAM à créer des instances de base de données dans un cluster de bases de données Neptune spécifié :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateInstance",
      "Effect": "Allow",
      "Action": "rds:CreateDBInstance",
      "Resource": "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster"
    }
  ]
}
```

Politique autorisant la création d'instances de base de données utilisant un groupe de paramètres de base de données spécifique

Voici un exemple de politique qui permet à un utilisateur IAM de créer des instances de base de données dans un cluster de bases de données Neptune spécifié (ici `us-west-2`) en utilisant uniquement un groupe de paramètres de base de données donné.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateInstanceWithPG",
      "Effect": "Allow",
      "Action": "rds:CreateDBInstance",
      "Resource": [
        "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster",
        "arn:aws:rds:us-west-2:123456789012:pg:my-instance-pg"
      ]
    }
  ]
}
```

Politique qui accorde l'autorisation de décrire n'importe quelle ressource

Voici un exemple de politique qui permet à un utilisateur IAM de décrire n'importe quelle ressource Neptune.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDescribe",
            "Effect": "Allow",
            "Action": "rds:Describe*",
            "Resource": "*"
        }
```

```
    ]
}
```

Tag-based exemples de politiques administratives

Les exemples suivants montrent comment créer des politiques administratives Neptune qui utilisent le balisage pour filtrer les autorisations pour diverses actions de gestion sur un cluster de bases de données.

Exemple 1 : accorder une autorisation pour des actions sur une ressource à l'aide d'une balise personnalisée qui peut prendre plusieurs valeurs

La politique ci-dessous autorise l'utilisation de l'API `ModifyDBInstance`, `CreateDBInstance` ou `DeleteDBInstance` sur toute instance de base de données dont la balise `env` est définie sur `dev` ou `test` :

JSON

```
{ "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDevTestAccess",
      "Effect": "Allow",
      "Action": [
        "rds:ModifyDBInstance",
        "rds:CreateDBInstance",
        "rds:DeleteDBInstance"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "rds:db-tag/env": [
              "dev",
              "test"
          ],
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Exemple 2 : limiter l'ensemble de clés et de valeurs de balise pouvant être utilisées pour baliser une ressource

Cette politique utilise une clé `Condition` pour autoriser une balise dotée de la clé `env` et de la valeur `test`, `qa` ou `dev` à être ajoutée à une ressource :

JSON

```
{ "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowTagAccessForDevResources",
      "Effect": "Allow",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "rds:req-tag/env": [
            "test",
            "qa",
            "dev"
          ],
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Exemple 3 : Autoriser l'accès complet aux ressources Neptune sur la base d'aws : ResourceTag

La politique suivante est similaire au premier exemple ci-dessus, mais utilise `aws:ResourceTag` à la place :

JSON

```
{ "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFullAccessToDev",
      "Effect": "Allow",
      "Action": [
        "rds:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/env": "dev",
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Création de déclarations de politique IAM personnalisées pour accéder aux données dans Amazon Neptune

Les déclarations de stratégie d'accès aux données Neptune utilisent des actions d'accès aux données, des ressources et des clés de condition, toutes précédées d'un préfixe `neptune-db:`.

Rubriques

- Utilisation d'actions de requête dans les déclarations de stratégie d'accès aux données Neptune
- Actions IAM pour l'accès aux données dans Amazon Neptune
- Types de ressources IAM pour accéder aux données dans Amazon Neptune
- Clés de condition IAM pour accéder aux données dans Amazon Neptune
- Création de politiques d'accès aux données IAM dans Amazon Neptune

Utilisation d'actions de requête dans les déclarations de stratégie d'accès aux données Neptune

Trois actions de requête Neptune peuvent être utilisées dans les déclarations de stratégie d'accès aux données, à savoir `ReadDataViaQuery`, `WriteDataViaQuery` et `DeleteDataViaQuery`. Une requête particulière peut nécessiter les autorisations nécessaires pour effectuer plusieurs de ces actions, et il n'est pas toujours évident de savoir quelle combinaison de ces actions doit être autorisée pour exécuter une requête.

Avant d'exécuter une requête, Neptune détermine les autorisations nécessaires pour exécuter chaque étape de la requête et les combine dans l'ensemble complet d'autorisations requises par la requête. Notez que cet ensemble complet d'autorisations inclut toutes les actions que la requête peut effectuer, ce qui n'est pas nécessairement l'ensemble des actions que la requête effectuera réellement lorsqu'elle traitera vos données.

Autrement dit, pour autoriser l'exécution d'une requête donnée, vous devez fournir les autorisations nécessaires pour chaque action que la requête est susceptible d'effectuer, qu'elle les exécute réellement ou non.

Voici quelques exemples de requêtes Gremlin qui permettent d'illustrer ce principe :

- ```
  g.V().count()
  ```

  `g.V()` et `count()` ne nécessitent qu'un accès en lecture. La requête ne nécessite donc qu'un accès `ReadDataViaQuery`.

- ```
  g.addV()
  ```

  `addV()` doit vérifier si un sommet avec un ID donné existe ou non avant d'en insérer un nouveau. La requête a donc besoin à la fois d'un accès `ReadDataViaQuery` et d'un accès `WriteDataViaQuery`.

- ```
  g.V('1').as('a').out('created').addE('createdBy').to('a')
  ```

  `g.V('1').as('a')` et `out('created')` ne nécessitent qu'un accès en lecture, mais `addE().from('a')` nécessite un accès en lecture et en écriture, car `addE()` doit lire les sommets `from` et `to` et vérifier si une arête portant le même ID existe déjà avant d'en ajouter une autre. La requête dans son ensemble a donc besoin à la fois d'un accès `ReadDataViaQuery` et d'un accès `WriteDataViaQuery`.
  ```

- ```
  g.V().drop()
  ```

  `g.V()` ne nécessite qu'un accès en lecture. `drop()` a besoin à la fois d'un accès en lecture et en suppression, car il doit lire un sommet ou une arête avant de le supprimer. La requête nécessite donc à la fois un accès `ReadDataViaQuery` et un accès `DeleteDataViaQuery`.

- ```
  g.V('1').property(single, 'key1', 'value1')
  ```

  `g.V('1')` ne nécessite qu'un accès en lecture, mais `property(single, 'key1', 'value1')` nécessite un accès en lecture, en écriture et en suppression. Ici, l'étape `property()` insère la clé et la valeur si elles n'existent pas déjà dans le sommet, mais si elles existent déjà, elle supprime la valeur de propriété existante et insère une nouvelle valeur à sa place. Par conséquent, la requête nécessite les accès `ReadDataViaQuery`, `WriteDataViaQuery` et `DeleteDataViaQuery`.

  Toute requête contenant une étape `property()` nécessite les autorisations `ReadDataViaQuery`, `WriteDataViaQuery` et `DeleteDataViaQuery`.

Voici quelques exemples openCypher :

- ```
  MATCH (n)
  RETURN n
  ```

  Cette requête lit tous les nœuds de la base de données et les renvoie, ce qui nécessite uniquement un accès `ReadDataViaQuery`.

- ```
  MATCH (n:Person)
  SET n.dept = 'AWS'
  ```

  Cette requête nécessite les accès `ReadDataViaQuery`, `WriteDataViaQuery` et `DeleteDataViaQuery`. Elle lit tous les nœuds portant l'étiquette « Personne » et y ajoute une nouvelle propriété avec la clé `dept` et la valeur `AWS`, ou si la propriété `dept` existe déjà, elle supprime l'ancienne valeur et insère `AWS` à la place. De même, si la valeur à définir est `null`, `SET` supprime complètement la propriété.

Étant donné que la clause SET peut dans certains cas nécessiter la suppression d'une valeur existante, elle a toujours besoin d'autorisations DeleteDataViaQuery, ainsi que d'autorisations ReadDataViaQuery et WriteDataViaQuery.

- ```
  MATCH (n:Person)
  DETACH DELETE n
  ```

  Cette requête nécessite les autorisations ReadDataViaQuery et DeleteDataViaQuery. Elle trouve tous les nœuds portant l'étiquette Person et les supprime ainsi que les arêtes connectées à ces nœuds et toutes les étiquettes et propriétés associées.

- ```
  MERGE (n:Person {name: 'John'})-[:knows]->(:Person {name: 'Peter'})
  RETURN n
  ```

  Cette requête nécessite les autorisations ReadDataViaQuery et WriteDataViaQuery. La clause MERGE correspond à un modèle spécifié ou le crée. Étant donné qu'une écriture peut avoir lieu si le modèle ne correspond pas, des autorisations d'écriture sont nécessaires, ainsi que des autorisations de lecture.

Actions IAM pour l'accès aux données dans Amazon Neptune

Notez que les actions d'accès aux données dans Neptune ont le préfixe neptune-db:, tandis que les actions administratives dans Neptune ont le préfixe rds:.

L'Amazon Resource Name (ARN) d'une ressource de données dans IAM n'est pas identique à l'ARN affecté au cluster au moment de la création. Vous devez construire l'ARN comme indiqué dans Spécification des ressources de données. L'ARN de ces ressources de données peut utiliser des caractères génériques pour inclure plusieurs ressources.

Data-access les instructions de politique peuvent également inclure la clé de QueryLanguage condition neptune-db : pour restreindre l'accès par langage de requête.

À compter de la Sortie : 1.2.0.0 (21/07/2022), Neptune prend en charge la restriction des autorisations pour une ou plusieurs actions Neptune spécifiques. Cela permet un contrôle d'accès plus précis qu'auparavant.

> **Important**
>
> - Il faut jusqu'à 10 minutes pour que les modifications apportées à une politique IAM s'appliquent aux ressources Neptune spécifiées.
> - Les politiques IAM appliquées à un cluster de bases de données Neptune s'appliquent à toutes les instances dans ce cluster.

Query-based actions d'accès aux données

> **Note**
>
> Les autorisations nécessaires pour exécuter une requête donnée ne sont pas toujours évidentes, car les requêtes peuvent effectuer diverses actions en fonction des données qu'elles traitent. Pour plus d'informatons, consultez Utilisation d'actions de requête.

`base de données Neptune : ReadDataViaQuery`

`ReadDataViaQuery` permet à l'utilisateur de lire les données de la base de données Neptune en soumettant des requêtes.

Groupes d'actions : lecture seule, lecture/écriture.

Clés de contexte d'action : `neptune-db:QueryLanguage`.

Ressources requises : base de données.

`base de données Neptune : WriteDataViaQuery`

`WriteDataViaQuery` permet à l'utilisateur d'écrire des données dans la base de données Neptune en soumettant des requêtes.

Groupes d'actions : lecture/écriture.

Clés de contexte d'action : `neptune-db:QueryLanguage`.

Ressources requises : base de données.

base de données Neptune : DeleteDataViaQuery

`DeleteDataViaQuery` permet à l'utilisateur de supprimer des données de la base de données Neptune en soumettant des requêtes.

Groupes d'actions : lecture/écriture.

Clés de contexte d'action : `neptune-db:QueryLanguage`.

Ressources requises : base de données.

base de données Neptune : GetQueryStatus

`GetQueryStatus` permet à l'utilisateur de vérifier le statut de toutes les requêtes actives.

Groupes d'actions : lecture seule, lecture/écriture.

Clés de contexte d'action : `neptune-db:QueryLanguage`.

Ressources requises : base de données.

base de données Neptune : GetStreamRecords

`GetStreamRecords` permet à l'utilisateur de récupérer des enregistrements de flux à partir de Neptune.

Groupes d'actions : lecture/écriture.

Clés de contexte d'action : `neptune-db:QueryLanguage`.

Ressources requises : base de données.

base de données Neptune : CancelQuery

`CancelQuery` permet à l'utilisateur d'annuler une requête.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

Actions générales d'accès aux données

base de données Neptune : GetEngineStatus

`GetEngineStatus` permet à l'utilisateur de vérifier le statut du moteur Neptune.

Groupes d'actions : lecture seule, lecture/écriture.

Ressources requises : base de données.

base de données Neptune : GetStatisticsStatus

`GetStatisticsStatus` permet à l'utilisateur de vérifier l'état des statistiques collectées pour la base de données.

Groupes d'actions : lecture seule, lecture/écriture.

Ressources requises : base de données.

base de données Neptune : GetGraphSummary

`GetGraphSummary` : l'API de résumé de graphe vous permet de récupérer un résumé en lecture seule de votre graphe.

Groupes d'actions : lecture seule, lecture/écriture.

Ressources requises : base de données.

base de données Neptune : ManageStatistics

`ManageStatistics` permet à l'utilisateur de gérer la collecte de statistiques pour la base de données.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

base de données Neptune : DeleteStatistics

`DeleteStatistics` permet à l'utilisateur de supprimer toutes les statistiques dans la base de données.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

base de données Neptune : ResetDatabase

`ResetDatabase` permet à l'utilisateur d'obtenir le jeton nécessaire à une réinitialisation et de réinitialiser la base de données Neptune.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

Bulk-loader actions d'accès aux données

`base de données Neptune : StartLoaderJob`

`StartLoaderJob` permet à l'utilisateur de démarrer une tâche de chargement en bloc.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

`base de données Neptune : GetLoaderJobStatus`

`GetLoaderJobStatus` permet à l'utilisateur de vérifier le statut d'une tâche de chargement en bloc.

Groupes d'actions : lecture seule, lecture/écriture.

Ressources requises : base de données.

`base de données Neptune : ListLoaderJobs`

`ListLoaderJobs` permet à l'utilisateur de répertorier toutes les tâches de chargement en bloc.

Groupes d'actions : affichage uniquement, lecture seule, lecture/écriture.

Ressources requises : base de données.

`base de données Neptune : CancelLoaderJob`

`CancelLoaderJob` permet à l'utilisateur d'annuler une tâche de chargement.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

Machine-learning actions d'accès aux données

`base de données Neptune : StartMLDataProcessingJob`

`StartMLDataProcessingJob` permet à un utilisateur de démarrer une tâche de traitement de données Neptune ML.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

base de données Neptune : StartMLModelTrainingJob

`StartMLModelTrainingJob` permet à un utilisateur de démarrer une tâche d'entraînement de modèle ML.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

base de données Neptune : StartMLModelTransformJob

`StartMLModelTransformJob` permet à un utilisateur de démarrer une tâche de transformation de modèle ML.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

Neptune-DB : créer un mLendpoint

`CreateMLEndpoint` permet à un utilisateur de créer un point de terminaison Neptune ML.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

base de données Neptune : GetMLDataProcessingJobStatus

`GetMLDataProcessingJobStatus` permet à un utilisateur de vérifier le statut d'une tâche de traitement de données Neptune ML.

Groupes d'actions : lecture seule, lecture/écriture.

Ressources requises : base de données.

base de données Neptune : GetMLModelTrainingJobStatus

`GetMLModelTrainingJobStatus` permet à un utilisateur de vérifier le statut d'une tâche d'entraînement de modèle Neptune ML.

Groupes d'actions : lecture seule, lecture/écriture.

Ressources requises : base de données.

base de données Neptune : GetMLModelTransformJobStatus

`GetMLModelTransformJobStatus` permet à un utilisateur de vérifier le statut d'une tâche de transformation de modèle Neptune ML.

Groupes d'actions : lecture seule, lecture/écriture.

Ressources requises : base de données.

base de données Neptune : GetMLEndpointStatus

`GetMLEndpointStatus` permet à un utilisateur de vérifier le statut d'un point de terminaison Neptune ML.

Groupes d'actions : lecture seule, lecture/écriture.

Ressources requises : base de données.

base de données Neptune : ListMLDataProcessingJobs

`ListMLDataProcessingJobs` permet à un utilisateur de répertorier toutes les tâches de traitement de données Neptune ML.

Groupes d'actions : affichage uniquement, lecture seule, lecture/écriture.

Ressources requises : base de données.

base de données Neptune : ListMLModelTrainingJobs

`ListMLModelTrainingJobs` permet à un utilisateur de répertorier toutes les tâches d'entraînement de modèle Neptune ML.

Groupes d'actions : affichage uniquement, lecture seule, lecture/écriture.

Ressources requises : base de données.

base de données Neptune : ListMLModelTransformJobs

`ListMLModelTransformJobs` permet à un utilisateur de répertorier toutes les tâches de transformation de modèle ML.

Groupes d'actions : affichage uniquement, lecture seule, lecture/écriture.

Ressources requises : base de données.

Neptune-DB : liste des points de mélange

`ListMLEndpoints` permet à un utilisateur de répertorier tous les points de terminaison Neptune ML.

Groupes d'actions : affichage uniquement, lecture seule, lecture/écriture.

Ressources requises : base de données.

base de données Neptune : CancelMLDataProcessingJob

`CancelMLDataProcessingJob` permet à un utilisateur d'annuler une tâche de traitement de données Neptune ML.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

base de données Neptune : CancelMLModelTrainingJob

`CancelMLModelTrainingJob` permet à un utilisateur d'annuler une tâche d'entraînement de modèle Neptune ML.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

base de données Neptune : CancelMLModelTransformJob

`CancelMLModelTransformJob` permet à un utilisateur d'annuler une tâche de transformation de modèle Neptune ML.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

Neptune-DB : Supprimer le point de repère

`DeleteMLEndpoint` permet à un utilisateur de supprimer un point de terminaison Neptune ML.

Groupes d'actions : lecture/écriture.

Ressources requises : base de données.

Types de ressources IAM pour accéder aux données dans Amazon Neptune

Les ressources de données, comme les actions sur les données, ont le préfixe `neptune-db:`.

Dans une stratégie d'accès aux données Neptune, vous spécifiez le cluster de bases de données auquel vous donnez accès dans un ARN au format suivant :

```
arn:aws:neptune-db:region:account-id:cluster-resource-id/*
```

Cet ARN de ressource contient les éléments suivants :

- *`region`*est la AWS région du cluster de base de données Amazon Neptune.
- *`account-id`* est le numéro de compte AWS du cluster de bases de données.
- *`cluster-resource-id`* est un ID de ressource pour le cluster de bases de données.

> **Important**
>
> L'`cluster-resource-id` est différent de l'identifiant du cluster. Pour trouver un ID de ressource de cluster dans le Neptune AWS Management Console, consultez la section Configuration du cluster de base de données en question.

Clés de condition IAM pour accéder aux données dans Amazon Neptune

À l'aide des clés de condition, vous pouvez spécifier des conditions dans une déclaration de politique IAM de sorte qu'elle ne prenne effet que lorsque les conditions sont remplies.

Les clés de condition que vous pouvez utiliser dans les déclarations de stratégie d'accès aux données Neptune appartiennent aux catégories suivantes :

- Clés de condition globales — Le sous-ensemble de clés de condition AWS globales que Neptune prend en charge dans les déclarations de politique d'accès aux données est répertorié ci-dessous.
- Service-specific clés de condition : il s'agit de clés définies par Neptune spécifiquement pour être utilisées dans les déclarations de politique d'accès aux données. À l'heure actuelle, il n'en existe qu'un seul, neptune-db : QueryLanguage, qui autorise l'accès uniquement si un langage de requête spécifique est utilisé.

AWS clés contextuelles de conditions globales prises en charge par Neptune dans les déclarations de politique d'accès aux données

Le tableau suivant répertorie le sous-ensemble de clés contextuelles de conditions globales AWS prises en charge par Amazon Neptune pour une utilisation dans les déclarations de stratégie d'accès aux données :

Clés de condition globales que vous pouvez utiliser dans les déclarations de stratégie d'accès aux données

Clés de condition	Description	Type
aws:CurrentTime	Filtre l'accès en fonction de la date et de l'heure actuelles de la demande.	String
aws:EpochTime	Filtre l'accès en fonction de la date et de l'heure de la demande, exprimées sous forme de valeur d'époque UNIX.	Numeric
aws:PrincipalAccount	Filtre l'accès en fonction du compte auquel appartient le principal à l'origine de la demande.	String
aws:PrincipalArn	Filtre l'accès via l'ARN du principal à l'origine de la demande.	String
aws:PrincipalIsAWSService	Autorise l'accès uniquement si l'appel est effectué directement par un principal AWS de service.	Boolean
aws:PrincipalOrgID	Filtre l'accès en fonction de l'identifiant de l'organisation dans AWS les Organizations auxquelles appartient le principal demandeur.	String
aws:PrincipalOrgPaths	Filtre l'accès par le chemin AWS Organizations pour le principal qui fait la demande.	String
aws:PrincipalTag	Filtre l'accès via une balise attachée au principal à l'origine de la demande.	String

Clés de condition	Description	Type
aws:PrincipalType	Filtre l'accès en fonction du type de principal à l'origine de la demande.	String
aws:RequestedRegion	Filtre l'accès en AWS fonction de la région appelée dans la demande.	String
aws:SecureTransport	Autorise l'accès uniquement si la demande a été envoyée en utilisant SSL.	Boolean
aws:SourceIp	Filtre l'accès en fonction de l'adresse IP du demandeur	String
aws:TokenIssueTime	Filtre l'accès selon la date et l'heure auxquelles les informations d'identification de sécurité temporaires ont été émises.	String
aws:UserAgent	Filtre l'accès par l'application cliente du demandeur.	String
aws:userid	Filtre l'accès en fonction de l'identifiant du principal du demandeur.	String
aws:ViaAWSService	Autorise l'accès uniquement si un AWS service a fait la demande en votre nom.	Boolean

Clés de condition spécifiques au service Neptune

Neptune prend en charge la clé de condition spécifique au service suivante pour les politiques IAM :

Clés de condition spécifiques au service Neptune

Clés de condition	Description	Type
neptune-db:QueryLanguage	Filtre l'accès aux données en fonction du langage de requête utilisé. Les valeurs valides sont Gremlin, OpenCypher et Sparql.	String

Clés de condition	Description	Type
	Les options prises en charge sont `ReadDataViaQuery` , `WriteDataViaQuery` , `DeleteDataViaQuery` , `GetQueryStatus` et `CancelQuery` .	

Création de politiques d'accès aux données IAM dans Amazon Neptune

Les exemples suivants montrent comment créer des politiques IAM personnalisées qui utilisent un contrôle d'accès précis aux API et aux actions du plan de données, depuis la version 1.2.0.0 du moteur Neptune.

Exemple de politique autorisant un accès illimité aux données d'un cluster de bases de données Neptune

L'exemple de politique suivant permet à un utilisateur IAM de se connecter à un cluster de bases de données à l'aide de l'authentification de base de données IAM. Il utilise le caractère * pour spécifier toutes les actions disponibles.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

L'exemple précédent inclut un ARN de ressource dans un format spécifique à l'authentification Neptune IAM. Pour créer l'ARN, consultez la section Spécification des ressources de données. Notez que l'ARN utilisée pour l'autorisation IAM `Resource` n'est pas le même que l'ARN assigné au cluster au moment de la création.

Exemple de politique autorisant l'accès en lecture seule à un cluster de bases de données Neptune

La politique suivante autorise l'accès complet en lecture seule aux données d'un cluster de bases de données Neptune :

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:Read*",
        "neptune-db:Get*",
        "neptune-db:List*"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Exemple de politique refusant tout accès à un cluster de bases de données Neptune

L'action IAM par défaut consiste à refuser l'accès à un cluster de bases de données, sauf si un effet *`Allow`* est accordé. Cependant, la politique suivante refuse tout accès à un cluster de base de données pour un AWS compte et une région particuliers, ce qui a alors la priorité sur tout `Allow` effet.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "neptune-db:*",
```

```
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Exemple de politique accordant un accès en lecture par le biais de requêtes

La politique suivante accorde uniquement l'autorisation de lire des données à partir d'un cluster de bases de données Neptune à l'aide d'une requête :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:ReadDataViaQuery",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Exemple de politique autorisant uniquement les requêtes Gremlin

La politique suivante utilise la clé de condition `neptune-db:QueryLanguage` pour autoriser Neptune uniquement avec le langage de requête Gremlin :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
```

```
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "neptune-db:QueryLanguage": "Gremlin"
        }
      }
    }
  ]
}
```

Exemple de politique autorisant tous les accès sauf l'accès à la gestion du modèle Neptune ML

La politique suivante accorde l'accès total aux opérations de graphes Neptune, à l'exception des fonctionnalités de gestion des modèles Neptune ML :

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:CancelLoaderJob",
        "neptune-db:CancelQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:DeleteStatistics",
        "neptune-db:GetEngineStatus",
        "neptune-db:GetLoaderJobStatus",
        "neptune-db:GetQueryStatus",
        "neptune-db:GetStatisticsStatus",
        "neptune-db:GetStreamRecords",
        "neptune-db:ListLoaderJobs",
        "neptune-db:ManageStatistics",
        "neptune-db:ReadDataViaQuery",
        "neptune-db:ResetDatabase",
        "neptune-db:StartLoaderJob",
        "neptune-db:WriteDataViaQuery"
      ],
```

```
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Exemple de politique autorisant l'accès à la gestion des modèles Neptune ML

Cette politique donne accès aux fonctionnalités de gestion des modèles Neptune ML :

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:CancelMLDataProcessingJob",
        "neptune-db:CancelMLModelTrainingJob",
        "neptune-db:CancelMLModelTransformJob",
        "neptune-db:CreateMLEndpoint",
        "neptune-db:DeleteMLEndpoint",
        "neptune-db:GetMLDataProcessingJobStatus",
        "neptune-db:GetMLEndpointStatus",
        "neptune-db:GetMLModelTrainingJobStatus",
        "neptune-db:GetMLModelTransformJobStatus",
        "neptune-db:ListMLDataProcessingJobs",
        "neptune-db:ListMLEndpoints",
        "neptune-db:ListMLModelTrainingJobs",
        "neptune-db:ListMLModelTransformJobs",
        "neptune-db:StartMLDataProcessingJob",
        "neptune-db:StartMLModelTrainingJob",
        "neptune-db:StartMLModelTransformJob"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Exemple de politique accordant un accès complet aux requêtes

La politique suivante accorde un accès complet aux opérations de requête de graphe Neptune, mais pas aux fonctionnalités telles que la réinitialisation rapide, les flux, le chargeur en bloc, la gestion des modèles Neptune ML, etc.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:GetEngineStatus",
        "neptune-db:GetQueryStatus",
        "neptune-db:CancelQuery"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Exemple de politique accordant un accès complet aux requêtes Gremlin uniquement

La politique suivante accorde un accès complet aux opérations de requête de graphe Neptune à l'aide du langage de requête Gremlin, mais pas aux requêtes dans les autres langages ni à des fonctionnalités telles que la réinitialisation rapide, les flux, le chargeur en bloc, la gestion des modèles Neptune ML, etc.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
```

```
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:GetEngineStatus",
        "neptune-db:GetQueryStatus",
        "neptune-db:CancelQuery"
      ],
      "Resource": [
        "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
      ],
      "Condition": {
        "StringEquals": {
           "neptune-db:QueryLanguage":"Gremlin"
        }
      }
    }
  ]
}
```

Exemple de politique accordant un accès complet à l'exception de la réinitialisation rapide

La politique suivante accorde l'accès total à un cluster de bases de données Neptune, à l'exception de l'utilisation de la réinitialisation rapide :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    },
    {
      "Effect": "Deny",
      "Action": "neptune-db:ResetDatabase",
```

```
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Utilisation de rôles liés à un service pour Amazon Neptune

Amazon Neptune utilise des rôles liés à un Gestion des identités et des accès AWS service (IAM). Un rôle lié à un service est un type unique de rôle IAM directement lié à Neptune. Service-linked les rôles sont prédéfinis par Neptune et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

> **Important**
>
> Pour certaines fonctionnalités de gestion, Amazon Neptune utilise une technologie opérationnelle qui est partagée avec Amazon RDS. Cela inclut les autorisations d'API de gestion et des rôles liés à un service.

Un rôle lié à un service simplifie l'utilisation de Neptune, car vous n'avez pas besoin d'ajouter manuellement les autorisations requises. Neptune définit les autorisations de ses rôles liés à un service. Sauf définition contraire, seul Neptune peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer les rôles uniquement après la suppression préalable de leurs ressources connexes. Vos ressources Neptune sont ainsi protégées, car vous ne pouvez pas involontairement supprimer l'autorisation d'accéder aux ressources.

Pour plus d'informations sur les autres services qui prennent en charge les rôles liés aux AWS services, consultez la section Services qui fonctionnent avec IAM et recherchez les services dont la valeur est Oui dans la Service-Linked colonne Rôle. Choisissez un Oui ayant un lien permettant de consulter les détails du rôle pour ce service.

Service-Linked Autorisations de rôle pour Neptune

Neptune utilise le rôle `AWSServiceRoleForRDS` lié à un service pour permettre à Neptune et Amazon RDS d' AWS appeler des services au nom de vos instances de base de données. Le rôle

lié à un service `AWSServiceRoleForRDS` fait confiance au service `rds.amazonaws.com` pour endosser le rôle.

La politique d'autorisations liée au rôle permet à Neptune de réaliser les actions suivantes au niveau des ressources spécifiées :

- Actions sur `ec2` :
  - `AssignPrivateIpAddresses`
  - `AuthorizeSecurityGroupIngress`
  - `CreateNetworkInterface`
  - `CreateSecurityGroup`
  - `DeleteNetworkInterface`
  - `DeleteSecurityGroup`
  - `DescribeAvailabilityZones`
  - `DescribeInternetGateways`
  - `DescribeSecurityGroups`
  - `DescribeSubnets`
  - `DescribeVpcAttribute`
  - `DescribeVpcs`
  - `ModifyNetworkInterfaceAttribute`
  - `RevokeSecurityGroupIngress`
  - `UnassignPrivateIpAddresses`
- Actions sur `sns` :
  - `ListTopic`
  - `Publish`
- Actions sur `cloudwatch` :
  - `PutMetricData`
  - `GetMetricData`
  - `CreateLogStream`
  - `PullLogEvents`
  - `DescribeLogStreams`
  - `CreateLogGroup`

> **Note**
>
> Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Il se peut que vous rencontriez le message d'erreur suivant :
> Impossible de créer la ressource. Vérifiez que vous détenez l'autorisation de créer un rôle lié au service. Dans le cas contraire, attendez et réessayez ultérieurement.
> Si vous voyez ce message, vérifiez que vous avez activé les autorisations suivantes :
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
> AWSServiceRoleForRDS",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Pour plus d'informations, consultez la section Autorisations des Service-Linked rôles dans le guide de l'utilisateur IAM.

Création d'un Service-Linked rôle pour Neptune

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez une instance ou un cluster, Neptune génère automatiquement le rôle lié au service.

> **Important**
>
> Pour plus d'informations, consultez Un nouveau rôle est apparu dans mon compte IAM dans le Guide de l'utilisateur IAM.

Si vous supprimez ce rôle lié à un service et que vous devez ensuite le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez une instance ou un cluster, à nouveau Neptune génère automatiquement le rôle lié au service.

Modifier un Service-Linked rôle pour Neptune

Neptune ne vous permet pas de modifier le rôle lié à un service `AWSServiceRoleForRDS`. Après avoir créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez la section Modification d'un Service-Linked rôle dans le guide de l'utilisateur IAM.

Supprimer un Service-Linked rôle pour Neptune

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez supprimer toutes vos instances et tous vos clusters avant de pouvoir supprimer le rôle lié à un service associé.

Nettoyage d'un Service-Linked rôle avant de le supprimer

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez d'abord vérifier qu'aucune session n'est active pour le rôle et supprimer toutes les ressources utilisées par le rôle.

Pour vérifier si une session est active pour le rôle lié à un service dans la console IAM

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Dans le panneau de navigation de la console IAM, choisissez Rôles. Ensuite, sélectionnez le nom (et non la case à cocher) du rôle `AWSServiceRoleForRDS`.
3. Sur la page Récapitulatif du rôle sélectionné, choisissez l'onglet Access Advisor.
4. Dans l'onglet Access Advisor, consultez l'activité récente pour le rôle lié à un service.

> **Note**
>
> Si vous n'êtes pas certain que Neptune utilise le rôle `AWSServiceRoleForRDS`, vous pouvez essayer de le supprimer. Si le service utilise le rôle, la suppression échoue et vous avez accès aux régions dans lesquelles le rôle est utilisé. Si le rôle est utilisé, vous devez attendre que la session se termine avant de pouvoir le supprimer. Vous ne pouvez pas révoquer la session d'un rôle lié à un service.

Si vous souhaitez supprimer le rôle `AWSServiceRoleForRDS`, vous devez commencer par supprimer toutes vos instances et tous vos clusters.

Suppression de toutes vos instances

Utilisez l'une des procédures suivantes pour supprimer chacune de vos instances.

Pour supprimer une instance (console)

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Instances.
3. Dans la liste Instances, choisissez l'instance que vous souhaitez supprimer.
4. Choisissez Actions d'instance), puis Supprimer.
5. Si vous on vous demande de Créer un aperçu final ?, sélectionnez Oui ou Non.
6. Si vous avez choisi Oui à l'étape précédente, dans le champ Nom de l'instantané final, saisissez le nom de votre instantané final.
7. Sélectionnez Delete (Supprimer).

Pour supprimer une instance (AWS CLI)

Consultez `delete-db-instance` dans la Référence de commande AWS CLI .

Pour supprimer une instance (API)

Consultez `DeleteDBInstance`.

Suppression de tous vos clusters

Utilisez l'une des procédures suivantes pour supprimer un seul cluster, puis répétez la procédure pour chacun de vos clusters.

Pour supprimer un cluster (console)

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans la liste Clusters, choisissez le cluster que vous souhaitez supprimer.
3. Choisissez Cluster Actions, puis Supprimer.

4. Sélectionnez Delete (Supprimer).

Pour supprimer un cluster (CLI)

Consultez `delete-db-cluster` dans la Référence de commande AWS CLI .

Pour supprimer un cluster (API)

Consultez `DeleteDBCluster`

Vous pouvez utiliser la console IAM, l'interface de ligne de commande IAM ou l'API IAM pour supprimer le rôle lié à un service `AWSServiceRoleForRDS`. Pour plus d'informations, consultez la section Suppression d'un Service-Linked rôle dans le guide de l'utilisateur IAM.

Utilisation d'informations d'identification temporaires pour se connecter à Amazon Neptune

Amazon Neptune prend en charge l'authentification IAM à l'aide d'informations d'identification temporaires.

Vous pouvez utiliser un rôle assumé pour vous authentifier à l'aide d'une politique d'authentification IAM, telles que l'un des exemples de politiques des sections précédentes.

Si vous utilisez des informations d'identification temporaires, vous devez spécifier `AWS_SESSION_TOKEN` en plus de `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` et `SERVICE_REGION`.

> **Note**
>
> Les informations d'identification temporaires expirent après un intervalle spécifique, y compris le jeton de session.
> Vous devez mettre à jour votre jeton de session lorsque vous demandez de nouvelles informations d'identification. Pour plus d'informations, consultez la section Utilisation d'informations d'identification de sécurité temporaires pour demander l'accès aux AWS ressources.

Les sections suivantes expliquent comment autoriser l'accès et récupérer des informations d'identification temporaires.

Pour vous authentifier à l'aide d'informations d'identification temporaires

1. Créez un rôle IAM avec l'autorisation d'accès à un cluster Neptune. Pour en savoir plus sur la création de ce rôle, consultez the section called “Types de politique IAM”.
2. Ajoutez au rôle une relation d'approbation qui autorise l'accès aux informations d'identification.

   Récupérez les informations d'identification temporaires, y compris `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` et `AWS_SESSION_TOKEN`.
3. Connectez-vous au cluster Neptune et signez les demandes à l'aide des informations d'identification temporaires. Pour plus d'informations sur la connexion et la signature de demandes, consultez the section called “Connexion avec l'authentification IAM”.

Vous pouvez récupérer des informations d'identification temporaires à l'aide de différentes méthodes en fonction de l'environnement.

Rubriques

- Obtenir des informations d'identification temporaires à l'aide du AWS CLI
- Configuration AWS Authentification Lambda pour Neptune IAM
- Configuration d'Amazon EC2 pour l'authentification IAM Neptune

Obtenir des informations d'identification temporaires à l'aide du AWS CLI

Pour obtenir des informations d'identification à l'aide de AWS Command Line Interface (AWS CLI), vous devez d'abord ajouter une relation de confiance qui autorise l' AWS utilisateur qui exécutera la AWS CLI commande à assumer le rôle.

Ajoutez la relation d'approbation suivante au rôle d'authentification IAM Neptune. Si vous ne disposez pas d'un rôle d'authentification IAM Neptune, consultez the section called “Types de politique IAM”.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "AWS": "arn:aws:iam::123456789012:user/test"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Pour plus d'informations sur l'ajout de la relation d'approbation au rôle, consultez Modification de la relation d'approbation pour un rôle existant dans le Guide d'administration d'AWS Directory Service .

Si la politique Neptune n'est pas encore attachée à un rôle, créez un rôle. Attachez la politique d'authentification IAM Neptune, puis ajoutez la politique d'approbation. Pour plus d'informations sur la création d'un nouveau rôle, consultez Création d'un rôle.

> **Note**
>
> Les sections suivantes supposent que vous avez AWS CLI installé le.

Pour exécuter le AWS CLI manuellement

1. Saisissez la commande suivante pour demander les informations d'identification à l'aide de l' AWS CLI. Remplacez l'ARN du rôle, le nom de la session et le profil par vos propres valeurs.

   ```
   aws sts assume-role  --role-arn  arn:aws:iam::123456789012:role/NeptuneIAMAuthRole
    --role-session-name test --profile testprofile
   ```

2. Voici un exemple de sortie de la commande. La section `Credentials` contient les valeurs dont vous avez besoin.

   > **Note**
   >
   > Notez la valeur du champ `Expiration`, car vous en aurez besoin pour obtenir de nouvelles informations d'identification après cette période.

   ```
   {
       "AssumedRoleUser": {
           "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:s3-access-example",
   ```

```
        "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-access-
example"
    },
    "Credentials": {
        "SecretAccessKey": "9drTJvcXLB89EXAMPLELB8923FB892xMFI",
        "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/qwjzP2iEXAMPLEbw/
m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8BRi
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=",
        "Expiration": "2016-03-15T00:05:07Z",
        "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
    }
}
```

3. Définissez les variables d'environnement à l'aide des informations d'identification renvoyées.

```
export AWS_ACCESS_KEY_ID=ASIAJEXAMPLEXEG2JICEA
export AWS_SECRET_ACCESS_KEY=9drTJvcXLB89EXAMPLELB8923FB892xMFI
export AWS_SESSION_TOKEN=AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/qwjzP2iEXAMPLEbw/
m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8BRi
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=

export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      ca-west-1 or sa-east-1 or eu-north-1 or eu-south-2 or eu-
west-1 or eu-west-2 or eu-west-3 or
                      eu-central-1 or me-south-1 or me-central-1 or il-central-1 or
 af-south-1 or ap-east-1 or
                      ap-northeast-1 or ap-northeast-2 or ap-northeast-3 or ap-
southeast-1 or ap-southeast-2 or
                      ap-southeast-3 or ap-southeast-4 or ap-southeast-5 or ap-
south-1 or ap-south-2 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

4. Connectez-vous en utilisant l'une des méthodes suivantes.

- the section called "Connexion à partir de la console Garmlin"
- the section called "Connexion à l'aide de Gremlin Java"
- the section called "Connexion avec SPARQL Java"
- the section called "Connexion avec Python"

Pour obtenir les informations d'identification à l'aide d'un script

1. Exécutez la commande suivante pour installer la commande jq. Le script utilise cette commande pour analyser le résultat de la AWS CLI commande.

```
sudo yum -y install jq
```

2. Créez un fichier nommé `credentials.sh` dans un éditeur de texte et ajoutez le texte suivant. Remplacez la région de service, l'ARN du rôle, le nom de la session et le profil par vos propres valeurs.

```
#!/bin/bash

creds_json=$(aws sts assume-role  --role-arn  arn:aws:iam::123456789012:role/
NeptuneIAMAuthRole --role-session-name test --profile testprofile)

export AWS_ACCESS_KEY_ID=$(echo "$creds_json" | jq .Credentials.AccessKeyId |tr -d
 '"')
export AWS_SECRET_ACCESS_KEY=$(echo "$creds_json" |
 jq .Credentials.SecretAccessKey| tr -d '"')
export AWS_SESSION_TOKEN=$(echo "$creds_json" | jq .Credentials.SessionToken|tr -d
 '"')

export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      ca-west-1 or sa-east-1 or eu-north-1 or eu-south-2 or eu-
west-1 or eu-west-2 or eu-west-3 or
                      eu-central-1 or me-south-1 or me-central-1 or il-central-1 or
 af-south-1 or ap-east-1 or
                      ap-northeast-1 or ap-northeast-2 or ap-northeast-3 or ap-
southeast-1 or ap-southeast-2 or
                      ap-southeast-3 or ap-southeast-4 or ap-southeast-5 or ap-
south-1 or ap-south-2 or
                      cn-north-1 or cn-northwest-1 or
```

```
                    us-gov-east-1 or us-gov-west-1
```

3. Connectez-vous en utilisant l'une des méthodes suivantes.

- the section called "Connexion à partir de la console Garmlin"
- the section called "Connexion à l'aide de Gremlin Java"
- the section called "Connexion avec SPARQL Java"
- the section called "Connexion avec Python"

Configuration AWS Authentification Lambda pour Neptune IAM

AWS Lambda inclut automatiquement les informations d'identification chaque fois que la fonction Lambda est exécutée.

Vous devez commencer par ajouter une relation d'approbation qui accorde au service Lambda l'autorisation d'endosser le rôle.

Ajoutez la relation d'approbation suivante au rôle d'authentification IAM Neptune. Si vous ne disposez pas d'un rôle d'authentification IAM Neptune, consultez the section called "Types de politique IAM".

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Pour plus d'informations sur l'ajout de la relation d'approbation au rôle, consultez Modification de la relation d'approbation pour un rôle existant dans le Guide d'administration d'AWS Directory Service.

Si la politique Neptune n'est pas encore attachée à un rôle, créez un rôle. Attachez la politique d'authentification IAM Neptune, puis ajoutez la politique d'approbation. Pour plus d'informations sur la création d'un nouveau rôle, consultez Création d'un rôle dans le Guide d'administration d'AWS Directory Service .

Pour accéder à Neptune à partir de Lambda

1. Connectez-vous à la AWS Lambda console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lambda/.
2. Créez une nouvelle fonction Lambda pour Python 3.6.
3. Attribuez le rôle `AWSLambdaVPCAccessExecutionRole` à la fonction Lambda. Cette action est requise pour accéder aux ressources Neptune qui sont accessibles par VPC uniquement.
4. Attribuez le rôle IAM d'authentification Neptune à la fonction Lambda.

   Pour plus d'informations, consultez Autorisations AWS Lambda dans le Guide du développeur AWS Lambda .
5. Copiez l'exemple Python d'authentification IAM dans le code de fonction Lambda.

   Pour plus d'informations sur l'exemple et l'exemple de code, consultez the section called "Connexion avec Python".

Configuration d'Amazon EC2 pour l'authentification IAM Neptune

Amazon EC2 vous permet d'utiliser des profils d'instance pour fournir automatiquement des informations d'identification. Pour plus d'informations, consultez Utilisation de profils d'instance dans le Guide de l'utilisateur IAM.

Vous devez commencer par ajouter une relation d'approbation qui accorde au service Amazon EC2 l'autorisation d'endosser le rôle.

Ajoutez la relation d'approbation suivante au rôle d'authentification IAM Neptune. Si vous ne disposez pas d'un rôle d'authentification IAM Neptune, consultez the section called "Types de politique IAM".

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Pour plus d'informations sur l'ajout de la relation d'approbation au rôle, consultez Modification de la relation d'approbation pour un rôle existant dans le Guide d'administration d'AWS Directory Service .

Si la politique Neptune n'est pas encore attachée à un rôle, créez un rôle. Attachez la politique d'authentification IAM Neptune, puis ajoutez la politique d'approbation. Pour plus d'informations sur la création d'un nouveau rôle, consultez Création d'un rôle dans le Guide d'administration d'AWS Directory Service .

Pour obtenir les informations d'identification à l'aide d'un script

1. Exécutez la commande suivante pour installer la commande jq. Le script utilise cette commande pour analyser la sortie de la commande curl.

   ```
   sudo yum -y install jq
   ```

2. Créez un fichier nommé `credentials.sh` dans un éditeur de texte et ajoutez le texte suivant. Remplacez la région de service par votre propre valeur.

   ```
   TOKEN=$( curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
   metadata-token-ttl-seconds: 21600" )
   role_name=$( curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/
   latest/meta-data/iam/security-credentials/ )
   creds_json=$( curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/
   latest/meta-data/iam/security-credentials/${role_name} )

   export AWS_ACCESS_KEY_ID=$(echo "$creds_json" | jq .AccessKeyId |tr -d '"')
   export AWS_SECRET_ACCESS_KEY=$(echo "$creds_json" | jq .SecretAccessKey| tr -d '"')
   export AWS_SESSION_TOKEN=$(echo "$creds_json" | jq .Token|tr -d '"')
   ```

```
export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      ca-west-1 or sa-east-1 or eu-north-1 or eu-south-2 or eu-
west-1 or eu-west-2 or eu-west-3 or
                      eu-central-1 or me-south-1 or me-central-1 or il-central-1 or
 af-south-1 or ap-east-1 or
                      ap-northeast-1 or ap-northeast-2 or ap-northeast-3 or ap-
southeast-1 or ap-southeast-2 or
                      ap-southeast-3 or ap-southeast-4 or ap-southeast-5 or ap-
south-1 or ap-south-2 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

3. Exécutez le script dans le shell `bash` à l'aide de la commande `source` :

```
source credentials.sh
```

   Une solution encore meilleure consiste à ajouter les commandes de ce script au fichier `.bashrc` de votre instance EC2 pour qu'elles soient appelées automatiquement lorsque vous vous connectez, ce qui rend les informations d'identification temporaires disponibles sur la console Gremlin.

4. Connectez-vous en utilisant l'une des méthodes suivantes.

   - the section called “Connexion à partir de la console Garmlin”
   - the section called “Connexion à l'aide de Gremlin Java”
   - the section called “Connexion avec SPARQL Java”
   - the section called “Connexion avec Python”

Enregistrement et surveillance de l'utilisation et des performances dans Amazon Neptune

Amazon Neptune prend en charge différentes méthodes pour surveiller les performances et l'utilisation :

- Statut de cluster : vérifiez l'état du moteur de base de données orienté graphe d'un cluster Neptune. Pour de plus amples informations, veuillez consulter the section called “Statut d'une instance”.

- Amazon CloudWatch — Neptune envoie automatiquement des métriques aux alarmes CloudWatch et les prend également en charge CloudWatch . Pour de plus amples informations, veuillez consulter the section called "En utilisant CloudWatch".
- Fichiers journaux d'audit : affichez, téléchargez ou consultez les fichiers journaux de base de données à l'aide de la console Neptune. Pour de plus amples informations, veuillez consulter the section called "Journaux d'audit avec Neptune".
- Publication de CloudWatch journaux sur Amazon Logs : vous pouvez configurer un cluster de base de données Neptune pour publier les données du journal d'audit dans un groupe de journaux dans Amazon CloudWatch Logs. Avec CloudWatch Logs, vous pouvez effectuer une analyse en temps réel des données du journal, l'utiliser CloudWatch pour créer des alarmes et afficher les métriques, et utiliser CloudWatch les journaux pour stocker vos enregistrements de journal dans un stockage hautement durable. Pour de plus amples informations, veuillez consulter Logs Neptune CloudWatch .
- AWS CloudTrail— Neptune prend en charge la journalisation des API en utilisant. CloudTrail Pour de plus amples informations, veuillez consulter the section called "Journalisation des appels d'API Neptune avec AWS CloudTrail".
- Balisage – Utilisez des balises pour ajouter des métadonnées à vos ressources Neptune et suivre l'utilisation en fonction des balises. Pour de plus amples informations, veuillez consulter the section called "Marquage des ressources Neptune".

Amazon Neptune et points de terminaison VPC d'interface (AWS PrivateLink)

Vous pouvez l'utiliser AWS PrivateLinkpour créer une connexion privée entre votre VPC et les points de terminaison de l'API Amazon Neptune. Vous pouvez accéder aux opérations de l'API Amazon Neptune comme si elles se trouvaient dans votre VPC, sans passer par une passerelle Internet, un appareil NAT, une connexion VPN ou une connexion. Le trafic entre votre VPC et Amazon Neptune reste au sein du réseau Amazon.

Amazon Neptune partage l'infrastructure d'API avec Amazon Relational Database Service. Pour configurer les points de terminaison VPC d'interface pour Amazon Neptune, suivez les instructions de l'API Amazon RDS et les points de terminaison VPC de l'interface (AWS PrivateLink) du guide de l'utilisateur Amazon Aurora.

Considérations relatives à la conformité pour Amazon Neptune

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. Pour plus d'informations sur votre responsabilité en matière de conformité lors de l'utilisation Services AWS, consultez AWS la documentation de sécurité.

Création de déploiements Amazon Neptune résilients et tolérants aux catastrophes

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone de disponibilité à l'autre sans interruption. Les zones de disponibilité sont plus hautement disponibles, tolérantes aux pannes et évolutives que les infrastructures traditionnelles à un ou plusieurs centres de données.

Un cluster de bases de données Amazon Neptune ne peut être créé que sur un réseau Amazon VPC comportant au moins deux sous-réseaux dans au moins deux zones de disponibilité. En répartissant vos instances de cluster sur deux zones de disponibilité au moins, Neptune garantit que des instances seront disponibles dans votre cluster de bases de données, dans l'éventualité peu probable d'une défaillance d'une zone de disponibilité. Le volume de votre cluster de bases de données Neptune couvre toujours trois zones de disponibilité afin d'offrir un stockage durable avec un risque moindre de perte des données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS mondiale.

Migration d'un graphe existant vers Amazon Neptune

Plusieurs outils et techniques peuvent vous aider à migrer des données de graphe existantes vers Amazon Neptune à partir d'un autre magasin de données.

Un flux de travail de migration simple implique les étapes suivantes :

- Exportez les données de leur magasin existant vers Amazon Simple Storage Service (Amazon S3).
- Nettoyez-les et mettez-les en forme pour l'importation.
- Chargez-les dans un cluster de bases de données Neptune à l'aide du Chargeur en bloc Neptune.
- Configurez votre application Gremlin ou SPARQL pour utiliser le point de terminaison correspondant fourni par Neptune.

> **Note**
>
> Le cluster Neptune doit être exécuté dans un VPC auquel votre application peut accéder.

Il existe des moyens de simplifier et d'automatiser certaines de ces étapes, en fonction de l'endroit où les données sont stockées :

Rubriques

- Migration de Neo4j vers Amazon Neptune
- Migration d'un graphe existant d'un serveur Apache TinkerPop Gremlin vers Amazon Neptune
- Migration d'un graphe existant d'un triplestore RDF vers Amazon Neptune
- Utilisation de AWS Database Migration Service (AWS DMS) pour migrer d'une base de données relationnelle ou NoSQL vers Amazon Neptune
- Migration de Blazegraph vers Amazon Neptune

Migration de Neo4j vers Amazon Neptune

Neo4j et Amazon Neptune sont des bases de données orientées graphe conçues pour les charges de travail de graphe transactionnelles en ligne qui prennent en charge le modèle de données de graphes à propriétés étiquetées. Ces similitudes font de Neptune un choix courant pour les clients qui cherchent à migrer leurs applications Neo4j actuelles. Toutefois, ces migrations ne se résument pas à un simple « lift and shift », car il existe des différences de langage, de prise en charge des fonctionnalités, de caractéristiques opérationnelles, d'architecture de serveur et de fonctionnalités de stockage entre les deux bases de données.

Cette page décrit le processus de migration et présente les éléments à prendre en compte avant de migrer une application orientée graphe Neo4j vers Neptune. Ces considérations s'appliquent généralement à toute application orientée graphe Neo4j, qu'elle soit alimentée par une base de données Community, Enterprise ou Aura. Bien que chaque solution soit unique et puisse nécessiter des procédures supplémentaires, toutes les migrations suivent le même schéma général.

Chacune des étapes décrites dans les sections suivantes inclut des considérations et des recommandations visant à simplifier le processus de migration. En outre, il existe des outils open source et des billets de blog décrivant le processus, ainsi qu'une section sur la compatibilité des fonctionnalités avec les options architecturales recommandées.

Rubriques

- Informations générales sur la migration de Neo4j vers Neptune
- Préparation à la migration de Neo4j vers Neptune
- Provisionnement de l'infrastructure lors de la migration de Neo4j vers Neptune
- Migration des données de Neo4j vers Neptune
- Migration d'une application de Neo4j vers Neptune
- Compatibilité de Neptune avec Neo4j
- Réécriture des requêtes Cypher pour les exécuter dans openCypher sur Neptune
- Ressources pour la migration de Neo4j vers Neptune

Informations générales sur la migration de Neo4j vers Neptune

Grâce à la prise en charge du langage de requête openCypher par Neptune, vous pouvez déplacer la plupart des charges de travail Neo4j qui utilisent le protocole Bolt ou HTTPS vers Neptune.

Cependant, openCypher est une spécification open source qui contient la plupart des fonctionnalités prises en charge par d'autres bases de données telles que Neo4j, mais pas toutes.

Bien qu'il soit compatible à bien des égards, Neptune ne remplace pas directement Neo4j. Neptune est un service de base de données orientée graphe entièrement géré et doté de fonctionnalités d'entreprise telles que la haute disponibilité et la haute durabilité. Son architecture est différente de celle de Neo4j. Neptune repose sur une instance, avec une seule instance d'enregistreur principale et jusqu'à 15 instances de réplica en lecture qui vous permettent de mettre à l'échelle horizontalement la capacité de lecture. Avec Neptune sans serveur, vous pouvez automatiquement augmenter ou diminuer la capacité de calcul en fonction du volume de requêtes. Cela est indépendant du stockage Neptune, qui est mis à l'échelle automatiquement à mesure que vous ajoutez des données.

Neptune prend en charge la spécification standard open source openCypher, version 9. Chez AWS, nous pensons que l'open source est bénéfique pour tout le monde et nous nous engageons à la fois à apporter la valeur de l'open source à nos clients et à apporter l'excellence opérationnelle AWS aux communautés open source.

Cependant, de nombreuses applications exécutées sur Neo4j utilisent également des fonctionnalités propriétaires qui ne sont pas open source et que Neptune ne prend pas en charge. Par exemple, Neptune ne prend pas en charge les procédures APOC, certaines clauses et fonctions spécifiques à Cypher ni les types de données `Char`, `Date` et `Duration`. Neptune convertit automatiquement les types de données manquants en types de données pris en charge.

Outre OpenCypher, Neptune prend également en charge le langage de requête Apache TinkerPop Gremlin pour les graphes de propriétés (ainsi que SPARQL pour les données RDF). Gremlin peut interagir avec openCypher sur le même graphe de propriétés, et dans de nombreux cas, vous pouvez utiliser Gremlin pour fournir des fonctionnalités qu'openCypher ne fournit pas. Vous trouverez ci-dessous une comparaison rapide des deux langages :

	openCypher	Gremlin
Style	Déclaratif	Impératif

Syntaxe	Mise en correspondance des modèles
	`Match p=(a)-[:route]->(d)`
	`WHERE a.code='ANC'`
		Basé sur la traversée
		`g.V().has('code', 'ANC').`
		`out('route').path().`

	openCypher	Gremlin
	`RETURN p`	`by(elementMap())`
Facilité d'utilisation	Inspiré de SQL, lisible par les non-programmeurs	Courbe d'apprentissage plus abrupte, similaire à celle des langages de programmation tels que Java
Flexibilité	Faible	Élevée
Prise en charge des requêtes	Requêtes basées sur des chaînes	Requêtes basées sur des chaînes ou code intégré pris en charge par les bibliothè ques clientes
Clients	HTTPS et Bolt	HTTPS et Websocket

En général, il n'est pas nécessaire de modifier votre modèle de données pour migrer de Neo4j vers Neptune, car Neo4j et Neptune prennent en charge les données de graphes de propriétés étiquetées (LPG). Neptune présente toutefois certaines différences d'architecture et de modèle de données dont vous pouvez tirer parti pour optimiser les performances. Par exemple :

- Neptune est traitée comme un IDs citoyen de premier ordre.
- Neptune utilise des politiques Gestion des identités et des accès AWS (IAM) pour sécuriser l'accès aux données de graphe de manière flexible et granulaire.
- Neptune propose plusieurs manières d'utiliser les blocs-notes Jupyter pour exécuter des requêtes et visualiser les résultats. Neptune fonctionne également avec des outils de visualisation tiers.
- Bien que Neptune n'ait aucun substitut à la bibliothèque Graph Data Science (GDS) de Neo4j, Neptune prend actuellement en charge l'analytique de graphe par le biais de diverses solutions. Par exemple, plusieurs exemples de blocs-notes montrent comment tirer parti de l'intégration de Neptune avec AWS le SDK Pandas dans les environnements Python pour exécuter des analyses sur des données graphiques.

N'hésitez pas à contacter l'équipe chargée AWS de votre AWS compte ou à contacter l'équipe chargée de votre compte si vous avez des questions. Nous tenons compte de vos commentaires pour traiter en priorité les nouvelles fonctionnalités qui répondront à vos besoins.

Préparation à la migration de Neo4j vers Neptune

La migration de la base de données de graphes Neo4j vers le service de base de données de graphes Neptune peut être abordée de deux manières principales : replateforme ou refactorisation/réarchitecture. L'approche de replateforme implique de modifier le modèle de données et l'architecture d'application existants afin de tirer le meilleur parti des capacités de Neptune, tandis que l'approche de refactorisation vise à trouver des composants équivalents dans Neptune afin de créer une implémentation comparable. Dans la pratique, une combinaison de ces stratégies est souvent utilisée, car le processus de migration consiste à équilibrer l'architecture Neptune cible avec les contraintes et les exigences de l'implémentation existante de Neo4j. Quelle que soit l'approche, l'essentiel est de repartir des cas d'utilisation de l'application pour concevoir le modèle de données, les requêtes et l'architecture globale qui répondent le mieux à vos besoins.

Approches en matière de migration

Lors de la migration d'une application Neo4j vers Neptune, nous recommandons l'une des deux stratégies suivantes : le changement de plateforme ou la refactorisation/restructuration de l'architecture. Pour plus d'informations sur les stratégies de migration, consultez le billet de blog 6 Strategies for Migrating Applications to the Cloud de Stephen Orban.

L'approche de replateforme, parfois appelée approche lift-tinker-and-shift, implique les étapes suivantes :

- Identifiez les cas d'utilisation auxquels votre application est censée répondre.
- Modifiez le modèle de données de graphe et l'architecture d'application existants pour répondre au mieux à ces besoins de charge de travail en utilisant les fonctionnalités de Neptune.
- Déterminez comment migrer les données, les requêtes et les autres parties de l'application source vers le modèle et l'architecture cibles.

Cette approche rétroactive vous permet de migrer votre application vers le type de solution Neptune que vous pourriez concevoir s'il s'agissait d'un tout nouveau projet.

L'approche de refactorisation, en revanche, implique les étapes suivantes :

- Identifiez les composants de la mise en œuvre existante, notamment l'infrastructure, les données, les requêtes et les fonctionnalités de l'application.
- Trouvez des équivalents dans Neptune qui pourront être utilisés pour créer une implémentation comparable.

Cette approche avant-gardiste vise à remplacer une implémentation par une autre.

Dans la pratique, il est probable que vous adoptiez une combinaison de ces deux approches. Vous pouvez commencer par un cas d'utilisation, concevoir l'architecture Neptune cible, puis vous tourner vers l'implémentation Neo4j existante pour identifier les contraintes et les invariants à respecter. Par exemple, vous devrez peut-être poursuivre l'intégration avec d'autres systèmes externes ou continuer à proposer des offres spécifiques APIs aux utilisateurs de votre application graphique. Grâce à ces informations, vous pourrez déterminer quelles données existantes transférer vers votre modèle cible, et celles qui doivent aller ailleurs.

À d'autres moments, vous commencerez peut-être par analyser un élément spécifique de votre implémentation Neo4j comme étant la meilleure source d'informations sur la tâche que votre application est censée accomplir. Ce type d'archéologie dans l'application existante peut aider à définir un cas d'utilisation que vous pourrez ensuite concevoir en utilisant les fonctionnalités de Neptune.

Que vous développiez une nouvelle application à l'aide de Neptune ou que vous migriez une application existante à partir de Neo4j, nous vous recommandons de travailler de manière rétroactive en partant des cas d'utilisation pour concevoir un modèle de données, un ensemble de requêtes et une architecture d'application répondant aux besoins de votre entreprise.

Différences architecturales entre Neptune et Neo4j

Lorsque les clients envisagent pour la première fois de migrer une application de Neo4j vers Neptune, il est souvent tentant d'effectuer une like-to-like comparaison basée sur la taille de l'instance. Cependant, les architectures de Neo4j et Neptune présentent des différences fondamentales. Neo4j est basé sur une all-in-one approche dans laquelle le chargement des données, l'ETL des données, les requêtes d'application, le stockage des données et les opérations de gestion se déroulent tous dans le même ensemble de ressources de calcul, telles que les instances EC2.

En revanche, Neptune est une base de données orientée graphe axée sur l'OLTP. Son architecture sépare les responsabilités, et ses ressources sont découplées afin qu'elles puissent se mettre à l'échelle de manière dynamique et indépendante.

Lors de la migration de Neo4j vers Neptune, déterminez les exigences de durabilité, de disponibilité et d'évolutivité des données de votre application. L'architecture de cluster de Neptune simplifie la conception d'applications nécessitant une durabilité, une disponibilité et une capacité de mise à l'échelle élevées. En comprenant l'architecture de cluster de Neptune, vous pourrez concevoir une topologie de cluster Neptune qui répond à ces exigences.

Architecture de cluster de Neo4j

De nombreuses applications de production utilisent le clustering causal de Neo4j pour garantir la durabilité, la haute disponibilité et la capacité de mise à l'échelle des données. L'architecture de clustering de Neo4j utilise des instances de serveur principal et de réplica en lecture :

- Les serveurs principaux assurent la durabilité des données et la tolérance aux pannes en répliquant les données à l'aide du protocole Raft.
- Les réplicas en lecture utilisent l'expédition des journaux de transactions pour répliquer les données de manière asynchrone pour les charges de travail à haut débit de lecture.

Chaque instance d'un cluster, qu'il s'agisse du serveur principal ou d'un réplica en lecture, contient une copie complète des données du graphe.

Architecture de cluster de Neptune

Un cluster Neptune est composé d'une instance d'enregistreur principale et d'un maximum de 15 instances de réplica en lecture. Toutes les instances du cluster partagent le même service de stockage distribué sous-jacent, distinct des instances.

- L'instance d'enregistreur principale coordonne toutes les opérations d'écriture dans la base de données et est évolutive verticalement pour assurer une prise en charge flexible des différentes charges de travail d'écriture. Elle prend également en charge les opérations de lecture.
- Les instances de réplica en lecture prennent en charge les opérations de lecture à partir du volume de stockage sous-jacent et vous permettent d'effectuer une mise à l'échelle horizontale pour répondre aux besoins des charges de travail à haut débit de lecture. Elles assurent également une haute disponibilité en servant de cibles de basculement pour l'instance principale.

  > **Note**
  >
  > Pour les charges de travail impliquant un grand nombre d'écritures, il est préférable de mettre à l'échelle les instances de réplica en lecture pour qu'elles soient de la même taille que l'instance d'enregistreur, afin de garantir la cohérence des lecteurs face aux modifications des données.

- Le volume de stockage sous-jacent met automatiquement à l'échelle la capacité de stockage à mesure que les données de votre base de données augmentent, jusqu'à 128 tébioctets (Tio) de stockage.

Les tailles d'instance sont dynamiques et indépendantes. Chaque instance peut être redimensionnée pendant l'exécution du cluster, et des réplicas en lecture peuvent être ajoutés ou supprimés pendant l'exécution du cluster.

La fonctionnalité Neptune sans serveur permet d'augmenter ou de diminuer automatiquement la capacité de calcul en fonction de la hausse ou de la baisse de la demande. Cela permet non seulement de réduire votre charge administrative, mais également de configurer la base de données pour qu'elle puisse gérer les pics de demande importants sans dégrader les performances ni vous obliger à provisionner plus de ressources que nécessaire.

Vous pouvez arrêter un cluster Neptune pendant une durée pouvant atteindre sept jours.

Neptune prend également en charge l'autoscaling afin d'ajuster automatiquement la taille des instances de lecteur en fonction de la charge de travail.

Grâce à la fonctionnalité de base de données globale de Neptune, vous pouvez mettre en miroir un cluster dans jusqu'à cinq autres régions.

Neptune est également tolérant aux pannes par conception :

- Le volume de cluster qui fournit le stockage de données à toutes les instances du cluster couvre plusieurs zones de disponibilité (AZs) en une seule Région AWS. Chaque zone de disponibilité contient une copie intégrale des données du cluster.
- Si l'instance principale devient indisponible, Neptune bascule automatiquement vers un réplica en lecture existant sans aucune perte de données, généralement en moins de 30 secondes. S'il n'existe aucune réplica en lecture dans le cluster, Neptune provisionne automatiquement une nouvelle instance principale, là aussi sans aucune perte de données.

Autrement dit, lors de la migration d'un cluster causal Neo4j vers Neptune, vous n'avez pas à concevoir explicitement la topologie du cluster pour une haute durabilité des données et une disponibilité élevée. Cela vous permet de dimensionner votre cluster en fonction des charges de travail de lecture et d'écriture attendues et de toute exigence de disponibilité accrue que vous pourriez avoir, en seulement quelques étapes :

- Pour mettre à l'échelle les opérations de lecture, ajoutez des instances de réplica en lecture ou activez la fonctionnalité Neptune sans serveur.
- Pour améliorer la disponibilité, distribuez l'instance principale et lisez les répliques dans votre cluster sur plusieurs zones de disponibilité (AZs).

- Pour réduire le temps de basculement, provisionnez au moins une instance de réplica en lecture qui servira de cible de basculement pour l'instance principale. Vous pouvez déterminer l'ordre dans lequel les réplicas en lecture sont promus en instance principale après un échec en affectant à chaque réplica une priorité. Il est recommandé de s'assurer qu'une cible de basculement possède une classe d'instances capable de gérer la charge de travail d'écriture de votre application si elle est promue en instance principale.

Différences de stockage des données entre Neptune et Neo4j

Neptune utilise un modèle de données de graphe basé sur un modèle de quadruplet natif. Lorsque vous migrez vos données vers Neptune, il existe plusieurs différences dont vous devez tenir compte dans l'architecture du modèle de données et de la couche de stockage afin d'utiliser de manière optimale le stockage partagé distribué et évolutif fourni par Neptune :

- Neptune n'utilise aucun schéma ni aucune contrainte définis de manière explicite. Il vous permet d'ajouter des nœuds, des arêtes et des propriétés de manière dynamique sans avoir à définir le schéma à l'avance. Neptune ne limite pas les valeurs et les types de données stockées, sauf indication contraire spécifiée dans les limites Neptune. Dans le cadre de l'architecture de stockage de Neptune, les données sont également automatiquement indexées de manière à gérer la plupart des modèles d'accès les plus courants. Cette architecture de stockage élimine les coûts opérationnels liés à la création et à la gestion du schéma de base de données et à l'optimisation des index.
- Neptune fournit une architecture de stockage distribuée et partagée unique qui s'adapte automatiquement par tranches de 10 Go à mesure que les besoins de stockage de votre base de données augmentent, jusqu'à 128 tébioctets (Tio). Cette couche de stockage est fiable, durable et tolérante aux pannes. Les données sont copiées six fois : deux fois dans chacune des trois zones de disponibilité. Elle fournit par défaut à tous les clusters Neptune une couche de stockage de données hautement disponible et tolérante aux pannes. L'architecture de stockage de Neptune réduit les coûts et évite d'avoir à provisionner du stockage ou plus de stockage que nécessaire pour faire face à la croissance future des données.

Avant de migrer vos données vers Neptune, il est conseillé de vous familiariser avec le modèle de données du graphe de propriétés et la sémantique des transactions de Neptune.

Différences opérationnelles entre Neptune et Neo4j

Neptune est un service entièrement géré qui automatise la plupart des tâches opérationnelles normales que vous devez effectuer lorsque vous utilisez des bases de données sur site ou autogérées telles que Neo4j Enterprise ou Community Edition :

- Sauvegardes automatisées : Neptune sauvegarde automatiquement le volume de votre cluster et conserve la sauvegarde pendant une période de rétention que vous spécifiez (comprise entre 1 et 35 jours). Ces sauvegardes étant continues et incrémentielles, vous pouvez rapidement opérer une restauration à un point quelconque de la période de rétention. Aucun impact sur les performances ou interruption du service de base de données ne se produit lors de l'écriture des données de sauvegarde.
- Instantanés manuels : Neptune vous permet de créer un instantané du volume de stockage de votre cluster de bases de données pour sauvegarder l'intégralité du cluster. Ce type d'instantané pourra être utilisé pour restaurer la base de données, en faire une copie et la partager entre les comptes.
- Clonage : Neptune prend en charge une fonctionnalité de clonage qui vous permet de créer rapidement des clones économiques d'une base de données. Les clones utilisent un copy-on-write protocole qui ne nécessite qu'un minimum d'espace supplémentaire après leur création. Le clonage de base de données est un moyen efficace de tester les nouvelles fonctionnalités ou mises à niveau de Neptune sans affecter le cluster d'origine.
- Surveillance : Neptune propose différentes méthodes pour surveiller les performances et l'utilisation de votre cluster, notamment :
  - Statut de l'instance
  - Intégration avec Amazon CloudWatch et AWS CloudTrail
  - Fonctionnalités du journal d'audit
  - Notifications d'événements
  - Identification
- Sécurité : Neptune fournit un environnement sécurisé par défaut. Un cluster réside dans un VPC privé qui permet d'isoler le réseau des autres ressources. Tout le trafic est crypté via SSL, et toutes les données sont cryptées au repos à l'aide de AWS KMS.

  En outre, Neptune s'intègre à Gestion des identités et des accès AWS (IAM) pour fournir une authentification. En spécifiant des clés de condition IAM, vous pouvez utiliser des politiques IAM pour fournir un contrôle d'accès précis au niveau des actions sur les données.

Différences d'outils et d'intégration entre Neptune et Neo4j

Neptune possède une architecture d'intégrations et d'outils différente de celle de Neo4j, ce qui peut avoir un impact sur l'architecture de votre application. Neptune utilise les ressources de calcul du cluster pour traiter les requêtes, mais utilise d'autres best-in-class AWS services pour des fonctionnalités telles que la recherche en texte intégral (utilisation OpenSearch), ETL (utilisation de Glue), etc. Pour obtenir la liste complète de ces intégrations, consultez Intégrations Neptune.

Provisionnement de l'infrastructure lors de la migration de Neo4j vers Neptune

Les clusters Amazon Neptune sont conçus pour évoluer en trois dimensions : stockage, capacité d'écriture et capacité de lecture. Les sections ci-dessous présentent les options spécifiques à prendre en compte lors de la migration.

Allocation du stockage

Le stockage de tout cluster Neptune est automatiquement provisionné, sans aucune charge administrative de votre part. Il est redimensionné dynamiquement par tranches de 10 Go à mesure que les besoins de stockage du cluster augmentent. Par conséquent, vous n'avez pas besoin d'estimer le stockage pour faire face à la croissance future des données ni de le provisionner, voire d'en provisionner plus que nécessaire.

Provisionnement de la capacité d'écriture

Neptune fournit une instance d'enregistreur unique qui peut être mise à l'échelle verticalement pour atteindre n'importe quelle taille d'instance disponible sur la page de tarification de Neptune. Lors de la lecture et de l'écriture de données dans une instance d'enregistreur, toutes les transactions sont conformes à l'ACID, avec un isolement des données tel que défini dans Niveaux d'isolement des transactions dans Neptune.

Le choix d'une taille optimale pour une instance services d'enregistreur nécessite l'exécution de tests de charge permettant de déterminer la taille d'instance optimale pour votre charge de travail. Toute instance dans Neptune peut être redimensionnée à tout moment en modifiant la classe d'instances de base de données. Vous pouvez estimer la taille d'une instance de départ en fonction de la simultanéité et de la latence moyenne des requêtes, comme décrit ci-dessous dans Estimation de la taille d'instance optimale lors du provisionnement du cluster.

Provisionnement de la capacité de lecture

Neptune est conçu pour mettre à l'échelle les instances de réplica en lecture à la fois horizontalement, en en ajoutant jusqu'à 15 au sein d'un cluster (ou plus dans une base de données globale Neptune), et verticalement selon la taille d'instance disponible sur la page de tarification de Neptune. Toutes les instances de réplica en lecture Neptune utilisent le même volume de stockage sous-jacent, ce qui permet une réplication transparente des données avec un retard minimal.

En plus de permettre la mise à l'échelle horizontale des demandes de lecture au sein d'un cluster Neptune, les réplicas en lecture servent également de cibles de basculement pour l'instance

d'enregistreur afin de garantir une haute disponibilité. Consultez Directives opérationnelles de base Amazon Neptune pour obtenir des suggestions sur la manière de déterminer le nombre et le placement appropriés des réplicas en lecture dans votre cluster.

Pour les applications où la connectivité et la charge de travail sont imprévisibles, Neptune prend également en charge une fonctionnalité d'autoscaling qui permet d'ajuster automatiquement le nombre de réplicas Neptune en fonction de critères que vous spécifiez.

Pour déterminer une taille et un nombre optimaux d'instances de réplication en lecture, il est nécessaire d'exécuter des tests de charge afin de déterminer les caractéristiques de la charge de travail de lecture qu'elles doivent prendre en charge. Toute instance dans Neptune peut être redimensionnée à tout moment en modifiant la classe d'instances de base de données. Vous pouvez estimer la taille d'une instance de départ en fonction de la simultanéité et de la latence moyenne des requêtes, comme décrit dans la section suivante.

Utilisation de Neptune sans serveur pour mettre automatiquement à l'échelle les instances de lecteur et d'enregistreur selon les besoins

Bien qu'il soit souvent utile d'estimer la capacité de calcul requise par les charges de travail prévues, vous pouvez configurer la fonctionnalité Neptune sans serveur pour augmenter ou diminuer automatiquement la capacité de lecture et d'écriture. Cela peut vous aider à répondre aux besoins en cas de hausse soudaine tout en réduisant automatiquement la capacité lorsque la demande diminue.

Estimation de la taille d'instance optimale lors du provisionnement du cluster

L'estimation de la taille d'instance optimale nécessite de connaître la latence moyenne des requêtes dans Neptune, lorsque votre charge de travail est exécutée, ainsi que le nombre de requêtes traitées simultanément. Une estimation approximative de la taille de l'instance peut être calculée en multipliant la latence moyenne des requêtes par le nombre de requêtes simultanées. Cela vous donne le nombre moyen de threads simultanés nécessaires pour gérer la charge de travail.

Chaque vCPU d'une instance Neptune peut prendre en charge deux threads de requête simultanés. En divisant les threads par 2, on obtient le nombre de v CPUs requis, qui peut ensuite être corrélé à la taille d'instance appropriée sur la page de tarification de Neptune. Par exemple :

```
Average Query Latency:         30ms (0.03s)
Number of concurrent queries:  1000/second

Number of threads needed:      0.03 x 1000 = 30 threads
```

```
Number of vCPUs needed:        30 / 2 = 15 vCPUs
```

En corrélant cela au nombre de v CPUs dans une instance, nous constatons que nous obtenons une estimation approximative du fait que a `r5.4xlarge` serait l'instance recommandée pour essayer cette charge de travail. Cette estimation est approximative et vise uniquement à fournir des indications initiales sur la sélection de la taille de l'instance. Toute application doit être soumise à un exercice de dimensionnement correct afin de déterminer le nombre et le ou les types d'instances appropriés pour la charge de travail.

Les besoins en mémoire doivent également être pris en compte, ainsi que les exigences de traitement. Neptune est particulièrement performant lorsque les données auxquelles les requêtes accèdent sont disponibles dans le cache du pool de mémoire tampon de la mémoire principale. Le provisionnement d'une mémoire suffisante peut également réduire les I/O coûts de manière significative.

Vous trouverez des informations et des conseils supplémentaires sur le dimensionnement des instances dans un cluster Neptune sur la page Dimensionnement des instances de base de données dans un cluster de bases de données Neptune.

Migration des données de Neo4j vers Neptune

Lorsque vous effectuez une migration de Neo4j vers Amazon Neptune, la migration des données constitue une étape majeure du processus. Plusieurs approches permettent de migrer des données. L'approche appropriée est déterminée par les besoins de l'application, la taille des données et le type de migration souhaité. Cependant, bon nombre de ces migrations nécessitent l'évaluation des mêmes considérations, dont plusieurs sont soulignées ci-dessous.

> **Note**
>
> Consultez la section Migration d'une base de données de graphes Neo4j vers Neptune à l'aide d'un utilitaire entièrement automatisé dans le blog de AWS base de données pour un exemple complet de migration de données hors ligne. step-by-step

Évaluation de la migration des données de Neo4j vers Neptune

Lors de l'évaluation d'une migration de données, la première étape consiste à déterminer la manière dont vous allez migrer les données. Les options dépendent de l'architecture de l'application à migrer, de la taille des données et des besoins de disponibilité pendant la migration. En général, les migrations se répartissent dans l'une des deux catégories suivantes : en ligne ou hors ligne.

Les migrations hors ligne sont généralement les plus simples à réaliser, car l'application n'accepte pas le trafic de lecture ou d'écriture pendant la migration. Une fois que l'application cesse d'accepter le trafic, les données peuvent être exportées, optimisées et importées, et l'application peut être testée avant sa réactivation.

Les migrations en ligne sont plus complexes, car l'application doit continuer à accepter le trafic de lecture et d'écriture pendant la migration des données. Les besoins exacts de chaque migration en ligne peuvent différer, mais l'architecture générale est souvent similaire à la suivante :

- Un flux des modifications continues apportées à la base de données doit être activé dans Neo4j en configurant Neo4j Streams comme source pour un cluster Kafka.
- Une fois cette opération terminée, une exportation du système en cours d'exécution peut être effectuée, en suivant les instructions fournies dans Exportation de données à partir de Neo4j lors de la migration vers Neptune, et l'heure peut être consignée pour une corrélation ultérieure avec la rubrique Kafka.

- Les données exportées sont ensuite importées dans Neptune, en suivant les instructions indiquées dans Importation de données à partir de Neo4j lors de la migration vers Neptune.
- Les données modifiées à partir du flux Kafka peuvent ensuite être copiées dans le cluster Neptune à l'aide d'une architecture similaire à celle décrite dans Writing to Amazon Neptune from Amazon Kinesis Data Streams. Notez que la réplication des modifications peut être exécutée en parallèle pour valider l'architecture et les performances de la nouvelle application.
- Une fois la migration des données validée, le trafic de l'application peut être redirigé vers le cluster Neptune, et l'instance Neo4j peut être mise hors service.

Optimisations des modèles de données pour la migration de Neo4j vers Neptune

Neptune et Neo4j prennent tous deux en charge les graphes de propriétés étiquetés (LPG). Neptune présente toutefois certaines différences d'architecture et de modèle de données dont vous pouvez tirer parti pour optimiser les performances :

Optimisation du nœud et du bord IDs

Neo4j génère automatiquement une longueur numérique. IDs Avec Cypher, vous pouvez faire référence aux nœuds par ID, mais cela est généralement déconseillé au profit d'une recherche des nœuds en fonction d'une propriété indexée.

Neptune vous permet de fournir votre propre chaîne pour les sommets et IDs les arêtes. Si vous ne fournissez pas les vôtres IDs, Neptune génère automatiquement des représentations sous forme de chaînes UUIDs pour les nouvelles arêtes et sommets.

Si vous migrez des données de Neo4j vers Neptune en les exportant depuis Neo4j puis en les important en masse dans Neptune, vous pouvez conserver celles de Neo4j. IDs Les valeurs numériques générées par Neo4j peuvent être fournies par l'utilisateur lors de l' IDs importation dans Neptune, où elles sont représentées sous forme de chaînes plutôt que de valeurs numériques.

Toutefois, dans certaines circonstances, il peut être utile de convertir une propriété de sommet en ID de sommet. Tout comme la recherche d'un nœud à l'aide d'une propriété indexée est le moyen le plus rapide de trouver un nœud dans Neo4j, la recherche d'un sommet par ID est le moyen le plus rapide de trouver un sommet dans Neptune. Par conséquent, si vous pouvez identifier une propriété de sommet appropriée contenant des valeurs uniques, envisagez de remplacer la valeur ~id du sommet par la valeur de propriété désignée dans les fichiers CSV de chargement en bloc. Dans ce cas, vous devrez également réécrire les valeurs d'arête ~from et ~to correspondantes dans les fichiers CSV.

Contraintes de schéma lors de la migration de données de Neo4j vers Neptune

Dans Neptune, la seule contrainte de schéma disponible est l'unicité de l'ID d'un nœud ou d'une arête. Les applications qui doivent utiliser une contrainte d'unicité sont invitées à envisager cette approche pour obtenir une contrainte d'unicité en spécifiant l'ID du nœud ou de l'arête. Si l'application utilisait plusieurs colonnes comme contrainte d'unicité, l'ID peut être défini sur une combinaison de ces valeurs. Par exemple, `id=123, code='SEA'` peut être représenté par`ID='123_SEA'`) pour obtenir une contrainte d'unicité complexe.

Optimisation de la direction des arêtes lors de la migration de données de Neo4j vers Neptune

Lorsque des nœuds, des arêtes ou des propriétés sont ajoutés à Neptune, ils sont automatiquement indexés de trois manières différentes, avec un quatrième index facultatif. En raison de la façon dont Neptune crée et utilise les index, les requêtes qui suivent les arêtes sortantes sont plus efficaces que celles qui utilisent des arêtes entrantes. En ce qui concerne le modèle de stockage de données de graphe de Neptune, il s'agit de recherches thématiques qui utilisent l'index SPOG.

Si, lors de la migration de votre modèle de données et de vos requêtes vers Neptune, vous constatez que vos requêtes les plus importantes reposent sur la traversée des arêtes entrantes où il existe un fort degré de distribution ramifiée, vous pouvez envisager de modifier votre modèle afin que ces traversées suivent plutôt les arêtes sortantes, en particulier lorsque vous ne pouvez pas spécifier les étiquettes d'arête à traverser. Pour ce faire, inversez la direction des arêtes concernées et mettez à jour les étiquettes des arêtes afin de refléter la sémantique de ce changement de direction. Par exemple, vous pouvez effectuer la modification suivante :

```
person_A – parent_of – person_B
   to:
person_B – child_of – person_A
```

Pour effectuer cette modification dans un fichier CSV d'arêtes à chargement en bloc, il suffit d'échanger les en-têtes de colonne `~from` et `~to`, et de mettre à jour les valeurs de la colonne `~label`.

Au lieu d'inverser la direction des arêtes, vous pouvez activer un quatrième index Neptune, l'index OSGP, qui rend la traversée des arêtes entrantes (ou les recherches basées sur des objets) beaucoup plus efficace. Cependant, l'activation de ce quatrième index réduit les taux d'insertion et nécessite davantage de stockage.

Optimisation du filtrage lors de la migration de données de Neo4j vers Neptune

Neptune est optimisé pour fonctionner au mieux lorsque les propriétés sont filtrées en fonction de la propriété la plus sélective disponible. Lorsque plusieurs filtres sont utilisés, l'ensemble de résultats correspondants est trouvé pour chacun de ces filtres, puis le chevauchement de tous les ensembles correspondants est calculé. Lorsque cela est possible, la combinaison de plusieurs propriétés en une seule permet de minimiser le nombre de recherches d'index et de réduire la latence d'une requête.

Par exemple, cette requête utilise deux recherches d'index et une jointure :

```
MATCH (n) WHERE n.first_name='John' AND n.last_name='Doe' RETURN n
```

Cette requête récupère les mêmes informations en utilisant une seule recherche d'index :

```
MATCH (n) WHERE n.name='John Doe' RETURN n
```

Neptune prend en charge différents types de données que Neo4j.

Mappages des types de données Neo4j dans les types de données pris en charge par Neptune

- Logique : `Boolean`

  Mappez cette valeur dans Neptune avec `Bool` ou `Boolean`.
- Numérique : `Number`

  Mappez cette valeur dans Neptune avec le type Neptune openCypher suivant le plus proche pouvant prendre en charge toutes les valeurs de la propriété numérique en question :

  ```
  Byte
  Short
  Integer
  Long
  Float
  Double
  ```
- Text : `String`

  Mappez cette valeur dans Neptune avec `String`.
- Point dans le temps :

```
Date
Time
LocalTime
DateTime
LocalDateTime
```

Mappez ces éléments dans Neptune avec `Date` au format UTC en utilisant l'un des formats ISO-8601 suivants pris en charge par Neptune :

```
yyyy-MM-dd
yyyy-MM-ddTHH:mm
yyyy-MM-ddTHH:mm:ss
yyyy-MM-ddTHH:mm:ssZ
```

- Durée : `Duration`

  Mappez cette valeur dans Neptune avec une valeur numérique pour l'arithmétique des dates, si nécessaire.

- Spatial : `Point`

  Mappez cette valeur dans Neptune avec les valeurs numériques des composants, chacune d'elles devenant alors une propriété distincte, ou exprimez-la sous forme de valeur de chaîne à interpréter par l'application cliente. Notez que l'intégration de la recherche en texte intégral de Neptune vous OpenSearch permet d'indexer les propriétés de géolocalisation.

Migration des propriétés à valeurs multiples de Neo4j vers Neptune

Neo4j permet de stocker des listes homogènes de types simples en tant que propriétés des nœuds et des arêtes. Ces listes peuvent contenir des valeurs en double.

Neptune n'autorise toutefois qu'une cardinalité définie ou unique pour les propriétés des sommets, et une cardinalité unique pour les propriétés des arêtes dans les données des graphes de propriétés. Par conséquent, il n'y a pas de migration directe des propriétés de liste de nœuds Neo4j contenant des valeurs dupliquées vers des propriétés de sommet Neptune ni de migration directe des propriétés de liste de relations Neo4j vers des propriétés d'arête Neptune.

Voici quelques stratégies possibles pour migrer les propriétés des nœuds à valeurs multiples Neo4j avec des valeurs en double dans Neptune :

- Supprimez les valeurs en double et convertissez la propriété du nœud Neo4j à valeurs multiples en propriété de sommet Neptune à cardinalité définie. Notez que l'ensemble Neptune peut ne pas refléter l'ordre des éléments dans la propriété Neo4j à valeurs multiples d'origine.
- Convertissez la propriété du nœud Neo4j à valeurs multiples en une représentation sous forme de chaîne d'une liste au format JSON dans une propriété de chaîne de sommet Neptune.
- Extrayez chacune des valeurs de propriété à valeurs multiples dans un sommet distinct doté d'une propriété de valeur, et connectez ces sommets au sommet parent à l'aide d'une arête portant le nom de la propriété.

De même, les stratégies possibles pour migrer les propriétés des relations Neo4j à valeurs multiples vers Neptune sont les suivantes :

- Convertissez la propriété de relation Neo4j à valeurs multiples en une représentation sous forme de chaîne d'une liste au format JSON et stockez-la en tant que propriété de chaîne d'arête Neptune.
- Refactorisez la relation Neo4j en arêtes Neptune entrantes et sortantes attachées à un sommet intermédiaire. Extrayez chacune des valeurs de propriété de relation à valeurs multiples dans un sommet distinct doté d'une propriété de valeur et connectez ces sommets au sommet intermédiaire à l'aide d'une arête portant le nom de la propriété.

Notez qu'une représentation sous forme de chaîne d'une liste au format JSON est opaque pour le langage de requête openCypher, bien qu'openCypher comprenne un prédicat `CONTAINS` qui permet des recherches simples dans des valeurs de chaîne.

Exportation de données à partir de Neo4j lors de la migration vers Neptune

Lorsque vous exportez des données à partir de Neo4j, utilisez les procédures APOC pour les exporter au format CSV ou GraphML. Bien qu'il soit possible d'exporter les données dans d'autres formats, il existe des outils open source pour convertir les données CSV exportées de Neo4j dans le format de chargement en bloc Neptune, ainsi que des outils open source pour convertir les données GraphML exportées de Neo4j dans le format de chargement en bloc Neptune.

Vous pouvez également exporter des données directement dans Amazon S3 à l'aide des différentes procédures APOC. L'exportation vers un compartiment Amazon S3 est désactivée par défaut, mais elle peut être activée à l'aide des procédures décrites dans la section Exportation dans Amazon S3 dans la documentation APOC de Neo4j.

Importation de données à partir de Neo4j lors de la migration vers Neptune

Vous pouvez importer des données dans Neptune en utilisant soit le chargeur en bloc Neptune, soit la logique de l'application dans un langage de requête compatible tel qu'openCypher.

Le chargeur en bloc Neptune est l'approche préférée pour importer de grandes quantités de données, car il fournit des performances d'importation optimisées si vous suivez les bonnes pratiques. Le chargeur en bloc prend en charge deux formats CSV différents, dans lesquels les données exportées à partir de Neo4j peuvent être converties à l'aide des utilitaires open source mentionnés ci-dessus dans la section Exportation de données.

Vous pouvez également utiliser openCypher pour importer des données avec une logique personnalisée pour l'analyse, la transformation et l'importation. Vous pouvez envoyer les requêtes openCypher soit via le point de terminaison HTTPS (ce qui est recommandé), soit en utilisant le pilote Bolt.

Migration d'une application de Neo4j vers Neptune

Après avoir migré vos données de Neo4j vers Neptune, l'étape suivante consiste à migrer l'application elle-même. Comme pour les données, plusieurs approches permettent de migrer votre application en fonction des outils que vous utilisez, des exigences, des différences architecturales, etc. Les éléments que vous devez généralement prendre en compte dans ce processus sont décrits ci-dessous.

Migration des connexions lors du passage de Neo4j à Neptune

Si vous n'utilisez pas actuellement les pilotes Bolt ou si vous souhaitez utiliser une alternative, vous pouvez vous connecter au point de terminaison HTTPS qui fournit un accès complet aux données renvoyées.

Si vous avez une application qui utilise le protocole Bolt, vous pouvez migrer ces connexions vers Neptune et laisser vos applications se connecter avec les mêmes pilotes que dans Neo4j. Pour vous connecter à Neptune, vous devrez peut-être apporter une ou plusieurs des modifications suivantes à votre application :

- L'URL et le port doivent être mis à jour pour utiliser les points de terminaison et le port du cluster (la valeur par défaut est 8182).
- Neptune exige que toutes les connexions utilisent le protocole SSL. Vous devez donc spécifier pour chaque connexion qu'elle est chiffrée.
- Neptune gère l'authentification par le biais de l'attribution de politiques et de rôles IAM. Les politiques et les rôles IAM fournissent un niveau de gestion des utilisateurs extrêmement flexible au sein de l'application. Il est donc important de lire et de comprendre les informations qui se trouvent dans la présentation d'IAM avant de configurer le cluster.
- Les connexions Bolt se comportent différemment dans Neptune que dans Neo4j à divers égards, comme expliqué dans Comportement des connexions Bolt dans Neptune.
- Vous trouverez plus d'informations et des suggestions dans Bonnes pratiques Neptune avec openCypher et Bolt.

Il existe des exemples de code pour les langages couramment utilisés tels que Java, Python, .NET et NodeJS, ainsi que pour des scénarios de connexion tels que l'utilisation de l'authentification IAM, dans Utilisation du protocole Bolt pour envoyer des requêtes openCypher à Neptune.

Acheminement des requêtes vers des instances de cluster lors du passage de Neo4j à Neptune

Les applications clientes Neo4j utilisent un pilote de routage et spécifient un mode d'accès pour acheminer les demandes de lecture et d'écriture vers un serveur approprié dans un cluster causal.

Lorsque vous migrez une application cliente vers Neptune, utilisez les points de terminaison Neptune pour acheminer efficacement les requêtes vers une instance appropriée du cluster :

- Toutes les connexions à Neptune doivent utiliser `bolt://` plutôt que `bolt+routing://` ou `neo4j://` dans l'URL.
- Le point de terminaison du cluster se connecte à l'instance principale actuelle de votre cluster. Utilisez le point de terminaison du cluster pour acheminer les demandes d'écriture vers l'instance principale.
- Le point de terminaison du lecteur distribue les connexions entre les instances de réplica en lecture du cluster. Si vous disposez d'un cluster à instance unique sans instance de réplica en lecture, le point de terminaison du lecteur se connecte à l'instance principale, qui prend en charge les opérations d'écriture. Si le cluster contient une ou plusieurs instances de réplica en lecture, l'envoi d'une demande d'écriture au point de terminaison du lecteur génère une exception.
- Chaque instance du cluster peut également avoir son propre point de terminaison d'instance. Utilisez un point de terminaison d'instance si votre application cliente doit envoyer une demande à une instance spécifique du cluster.

Pour de plus amples informations, veuillez consulter Considérations relatives aux points de terminaison Neptune.

Cohérence des données dans Neptune

Lorsque vous utilisez des clusters causaux Neo4j, les réplicas en lecture sont cohérents à terme avec les serveurs principaux, mais les applications clientes peuvent garantir la cohérence causale en utilisant le chaînage causal. Le chaînage causal implique le transfert de signets entre les transactions, ce qui permet à une application cliente d'écrire sur un serveur principal, puis de lire sa propre écriture à partir d'un réplica en lecture.

Dans Neptune, les instances de réplica en lecture sont cohérents à terme avec l'enregistreur, avec un retard de réplication généralement inférieur à 100 millisecondes. Toutefois, tant qu'une modification n'a pas été répliquée, les mises à jour des arêtes et sommets existants, ainsi que les ajouts d'arêtes

et de sommets ne sont pas visibles sur une instance de réplica. Par conséquent, si votre application a besoin d'une cohérence immédiate sur Neptune en lisant chaque écriture, utilisez le point de terminaison du cluster pour l' read-after-writeopération. Il s'agit du seul moment où vous pouvez utiliser le point de terminaison de cluster pour les opérations de lecture. Dans tous les autres cas, utilisez le point de terminaison du lecteur pour les lectures.

Migration des requêtes de Neo4j vers Neptune

Bien que la prise en charge d'openCypher par Neptune réduise considérablement la quantité de travail requise pour migrer des requêtes à partir de Neo4j, certaines différences restent à évaluer lors de la migration :

- Comme indiqué dans Optimisations des modèles de données ci-dessus, vous devrez peut-être apporter des modifications à votre modèle de données afin de créer un modèle de données de graphe optimisé pour Neptune, ce qui impliquera aussi des modifications de vos requêtes et de vos tests.
- Neo4j propose une variété d'extensions de langage spécifiques à Cypher, qui ne sont pas incluses dans la spécification openCypher implémentée par Neptune. Selon le cas d'utilisation et la fonctionnalité utilisés, il peut y avoir des solutions de contournement dans le langage openCypher, en utilisant le langage Gremlin ou en utilisant d'autres mécanismes tels que décrits dans Réécriture des requêtes Cypher pour les exécuter dans openCypher sur Neptune.
- Les applications utilisent souvent d'autres composants intergiciels pour interagir avec la base de données au lieu des pilotes Bolt eux-mêmes. Vérifiez Compatibilité de Neptune avec Neo4j pour déterminer si les outils ou intergiciels que vous utilisez sont pris en charge.
- En cas de basculement, le pilote Bolt peut continuer à se connecter à l'instance d'enregistreur ou de lecteur précédente, car le point de terminaison du cluster fourni pour la connexion a été résolu en adresse IP. La gestion appropriée des erreurs dans votre application devrait permettre de résoudre ce problème, comme décrit dans Création d'une connexion après un basculement.
- Lorsque des transactions sont annulées en raison de conflits non résolus ou de délais d'attente de verrouillage, Amazon Neptune répond avec une exception `ConcurrentModificationException`. Pour de plus amples informations, veuillez consulter Codes d'erreur du moteur. En tant que bonne pratique, les clients doivent toujours intercepter et gérer ces exceptions.

  Une exception `ConcurrentModificationException` se produit parfois lorsque plusieurs threads ou plusieurs applications écrivent simultanément sur le système. En raison des niveaux d'isolement des transactions, ces conflits sont parfois inévitables.

- Neptune prend en charge l'exécution des requêtes Gremlin et openCypher au niveau des mêmes données. En d'autres termes, dans certains scénarios, vous devrez peut-être envisager d'utiliser Gremlin, avec ses fonctionnalités de requête plus puissantes, pour exécuter certaines des fonctionnalités de vos requêtes.

Comme indiqué dans Provisionnement de l'infrastructure ci-dessus, chaque application doit être soumise à un exercice de dimensionnement correct afin de garantir que le nombre d'instances, la taille des instances et la topologie du cluster sont tous optimisés pour la charge de travail spécifique de l'application.

Les considérations abordées ici pour la migration de votre application sont les plus courantes, mais cette liste n'est pas exhaustive. Chaque application est unique. N'hésitez pas à contacter l'équipe AWS chargée de votre compte ou à contacter l'équipe chargée de votre compte si vous avez d'autres questions.

Fonctionnalités et outils de migration spécifiques à Neo4j

Neo4j possède une variété de fonctionnalités personnalisées et de modules complémentaires avec des fonctionnalités sur lesquelles votre application peut s'appuyer. Lors de l'évaluation de la nécessité de migrer cette fonctionnalité, il est souvent utile de déterminer s'il existe une meilleure approche AWS pour atteindre le même objectif. Compte tenu des différences architecturales entre Neo4j et Neptune, vous pouvez souvent trouver des alternatives efficaces qui tirent parti d'autres AWS services ou intégrations.

Consultez Compatibilité de Neptune avec Neo4j pour obtenir la liste des fonctionnalités spécifiques à Neo4J et des solutions de contournement suggérées.

Compatibilité de Neptune avec Neo4j

Neo4j adopte une approche all-in-one architecturale, dans laquelle le chargement des données, l'ETL des données, les requêtes d'application, le stockage des données et les opérations de gestion se produisent tous dans le même ensemble de ressources de calcul, telles que EC2 les instances. Amazon Neptune est une base de données orientée graphe avec des spécifications ouvertes et axée sur l'OLTP. Son architecture sépare les opérations et dissocie les ressources afin qu'elles puissent se mettre à l'échelle de manière dynamique.

Neo4j contient une variété de fonctionnalités et d'outils, y compris des outils tiers, qui ne font pas partie de la spécification openCypher, qui sont incompatibles avec openCypher ou qui sont incompatibles avec l'implémentation Neptune d'openCypher. Vous trouverez ci-dessous une liste des plus courants.

Fonctionnalités spécifiques à Neo4J qui ne se trouvent pas dans Neptune

- **`LOAD CSV`** : Neptune a une approche architecturale différente de celle de Neo4j pour charger les données. Pour permettre une meilleure mise à l'échelle et une meilleure optimisation des coûts, Neptune met en œuvre une séparation des problèmes relatifs aux ressources et recommande d'utiliser l'une des intégrations de services AWS (par exemple, AWS Glue) afin d'exécuter les processus ETL requis pour préparer les données dans un format pris en charge par le chargeur en bloc Neptune.

  Une autre option consiste à faire la même chose en utilisant du code d'application exécuté sur des ressources de AWS calcul telles que les EC2 instances Amazon, les fonctions Lambda, Amazon Elastic Container Service, les AWS Batch tâches, etc. Le code peut utiliser le point de terminaison HTTPS de Neptune ou le point de terminaison Bolt.
- Contrôle d'accès précis : Neptune prend en charge le contrôle d'accès granulaire des actions d'accès aux données à l'aide de clés de condition IAM. Un contrôle d'accès précis supplémentaire peut être mis en œuvre au niveau de la couche application.
- Neo4j Fabric : Neptune prend en charge la fédération de requêtes entre les bases de données pour les charges de travail RDF à l'aide du mot clé SPARQL `SERVICE`. Comme il n'existe pas actuellement de norme ou de spécification ouverte pour la fédération de requêtes pour les charges de travail des graphes de propriétés, cette fonctionnalité devrait être implémentée au niveau de la couche application.
- Contrôle d'accès basé sur les rôles (RBAC) : Neptune gère l'authentification en attribuant des politiques et des rôles IAM. Les politiques et les rôles IAM fournissent un niveau de gestion des

utilisateurs extrêmement flexible au sein d'une application. Il est donc utile de lire et de comprendre les informations qui se trouvent dans la présentation d'IAM avant de configurer le cluster.

- Signets : les clusters Neptune se composent d'une seule instance d'enregistreur et de jusqu'à 15 instances de réplica en lecture. Les données écrites dans l'instance d'enregistreur sont conformes à la norme ACID et offrent une solide garantie de cohérence lors des lectures ultérieures. Les réplicas en lecture utilisent le même volume de stockage que l'instance d'enregistreur et sont cohérentes à terme, généralement en moins de 100 ms à partir du moment où les données sont écrites. Si votre cas d'utilisation nécessite immédiatement de garantir la cohérence de lecture des nouvelles écritures, ces lectures doivent être dirigées vers le point de terminaison du cluster plutôt que vers le point de terminaison du lecteur.
- Procédures APOC étant donné que les procédures APOC ne sont pas incluses dans la spécification openCypher, Neptune ne fournit pas de prise en charge directe pour les procédures externes. Neptune s'appuie plutôt sur des intégrations avec d'autres AWS services pour obtenir des fonctionnalités similaires pour les utilisateurs finaux de manière évolutive, sécurisée et robuste. Parfois, les procédures APOC peuvent être réécrites dans openCypher ou Gremlin, et certaines ne sont pas pertinentes pour les applications Neptune.

  En général, les procédures APOC entrent dans les catégories ci-dessous :

  - Importation : Neptune prend en charge l'importation de données dans divers formats à l'aide de langages de requête, du chargeur en bloc Neptune ou en tant que cible d'AWS Database Migration Service. Les opérations ETL sur les données peuvent être effectuées à AWS Glue l'aide du package `neptune-python-utils`open source.
  - Exportation : Neptune prend en charge l'exportation de données à l'aide de l'utilitaire `neptune-export`, qui accepte une variété de formats et de méthodes d'exportation courants.
  - Intégration de base de données — Neptune prend en charge l'intégration avec d'autres bases de données à l'aide d'outils ETL tels que AWS Glue ou d'outils de migration tels que. AWS Database Migration Service
  - Mises à jour des graphes : Neptune propose un ensemble complet de fonctionnalités permettant de mettre à jour les données des graphes de propriétés grâce à sa prise en charge des langages de requête openCypher et Gremlin. Consultez Réécritures Cypher pour obtenir des exemples de réécritures de procédures couramment utilisées.
  - Structures de données : Neptune propose un ensemble complet de fonctionnalités permettant de mettre à jour les données des graphes de propriétés grâce à sa prise en charge des langages de requête openCypher et Gremlin. Consultez Réécritures Cypher pour obtenir des exemples de réécritures de procédures couramment utilisées.

  - Temporel (date et heure) : Neptune propose un ensemble complet de fonctionnalités pour mettre à jour les données du graphe de propriétés grâce à sa prise en charge des langages de requête openCypher et Gremlin. Consultez Réécritures Cypher pour obtenir des exemples de réécritures de procédures couramment utilisées.
  - Mathématique : Neptune propose un ensemble complet de fonctionnalités permettant de mettre à jour les données des graphes de propriétés grâce à sa prise en charge des langages de requête openCypher et Gremlin. Consultez Réécritures Cypher pour obtenir des exemples de réécritures de procédures couramment utilisées.
  - Requête de graphe avancée : Neptune propose un ensemble complet de fonctionnalités pour mettre à jour les données des graphes de propriétés grâce à sa prise en charge des langages de requête openCypher et Gremlin. Consultez Réécritures Cypher pour obtenir des exemples de réécritures de procédures couramment utilisées.
  - Comparaison de graphes : Neptune propose un ensemble complet de fonctionnalités pour mettre à jour les données des graphes de propriétés grâce à sa prise en charge des langages de requête openCypher et Gremlin. Consultez Réécritures Cypher pour obtenir des exemples de réécritures de procédures couramment utilisées.
  - Exécution Cypher : Neptune prend en charge un ensemble complet de fonctionnalités pour mettre à jour les données du graphe de propriétés grâce à sa prise en charge des langages de requête openCypher et Gremlin. Consultez Réécritures Cypher pour obtenir des exemples de réécritures de procédures couramment utilisées.
- Procédures personnalisées : Neptune ne prend pas en charge les procédures personnalisées créées par les utilisateurs. Cette fonctionnalité devrait être implémentée au niveau de la couche application.
- Géospatiale — Bien que Neptune ne fournisse pas de support natif pour les fonctionnalités géospatiales, des fonctionnalités similaires peuvent être obtenues grâce à l'intégration à AWS d'autres services, comme le montre ce billet de blog : Combinez Amazon Neptune et OpenSearch Amazon Service pour les requêtes géospatiales de Ross Gabay et Abhilash Vinod (1er février 2022).
- Science des données graphiques : Neptune prend aujourd'hui en charge l'analyse de graphes par le biais de Neptune Analytics, un moteur optimisé pour la mémoire qui prend en charge une bibliothèque d'algorithmes d'analyse de graphes.

  Neptune propose aussi une intégration avec le kit SDK AWS Pandas et plusieurs exemples de blocs-notes qui montrent comment tirer parti de cette intégration dans les environnements Python pour exécuter des analyses sur des données de graphe.

- Contraintes de schéma : dans Neptune, la seule contrainte de schéma disponible est l'unicité de l'ID d'un nœud ou d'une arête. Aucune fonctionnalité ne permet de spécifier d'autres contraintes de schéma ni de contraintes d'unicité ou de valeur supplémentaires sur un élément du graphe. Les valeurs d'ID dans Neptune sont des chaînes et peuvent être définies à l'aide de Gremlin, comme ceci :

  ```
  g.addV('person').property(id, '1') )
  ```

  Les applications qui doivent utiliser l'ID comme contrainte d'unicité sont encouragées à essayer cette approche pour obtenir une contrainte d'unicité. Si l'application utilisait plusieurs colonnes comme contrainte d'unicité, l'ID peut être défini sur une combinaison de ces valeurs. Par exemple, `id=123, code='SEA'` pourrait être représenté comme `ID='123_SEA'` pour obtenir une contrainte d'unicité complexe.

- Architecture mutualisée : Neptune ne prend en charge qu'un seul graphe par cluster. Pour créer un système multilocataire à l'aide de Neptune, utilisez plusieurs clusters ou partitionnez logiquement les locataires au sein d'un seul graphe et appliquez la séparation avec la logique côté application. Par exemple, ajoutez une propriété `tenantId` et incluez-la dans chaque requête, comme suit :

  ```
  MATCH p=(n {tenantId:1})-[]->({tenantId:1}) RETURN p LIMIT 5)
  ```

  Neptune sans serveur facilite la mise en œuvre de l'architecture mutualisée à l'aide de plusieurs clusters de bases de données, chacun étant mis à l'échelle indépendamment et automatiquement selon les besoins.

Prise en charge de Neptune pour les outils Neo4j

Neptune propose les alternatives suivantes aux outils Neo4j :

- Navigateur Neo4j : Neptune offre des blocs-notes de graphe open source qui fournissent un IDE axé sur les développeurs pour exécuter des requêtes et visualiser les résultats.
- Neo4j Bloom : Neptune prend en charge les visualisations de graphe enrichies à l'aide de solutions de visualisation tierces telles que Graph-explorer, Tom Sawyer, Cambridge Intelligence, Graphistry, metaphacts et G.V().
- GraphQL — Neptune prend actuellement en charge GraphQL par le biais d'intégrations personnalisées. AWS AppSync Consultez le billet de blog Build a graph application with Amazon

Neptune and AWS Amplify, et l'exemple de projet intitulé Building Serverless Calorie Tracker application with and Amazon Neptune. AWS AppSync

- NeoSemantics— Neptune prend en charge le modèle de données RDF de manière native. Il est donc conseillé aux clients souhaitant exécuter des charges de travail RDF d'utiliser le support du modèle RDF de Neptune.
- Arrows.app : le chiffrement créé lors de l'exportation du modèle à l'aide de la commande d'exportation est compatible avec Neptune.
- Linkurious Ogma : un exemple d'intégration avec Linkurious Ogma est disponible ici.
- Spring Data Neo4j : n'est actuellement pas compatible avec Neptune.
- Connecteur Neo4j Spark : le connecteur Neo4j Spark peut être utilisé dans une tâche Spark pour se connecter à Neptune à l'aide d'openCypher. Voici un exemple de code et un exemple de configuration d'application :

  Exemple de code :

```
SparkSession spark = SparkSession
            .builder()
            .config("encryption.enabled", "true")
            .appName("Simple Application").config("spark.master",
 "local").getOrCreate();

Dataset<Row> df = spark.read().format("org.neo4j.spark.DataSource")
            .option("url", "bolt://(your cluster endpoint):8182")
            .option("encryption.enabled", "true")
            .option("query", "MATCH (n:airport) RETURN n")
            .load();

System.out.println("TOTAL RECORD COUNT: " + df.count());
spark.stop();
```

  Configuration de l'application :

```
<dependency>
    <groupId>org.neo4j</groupId>
    <artifactId>neo4j-connector-apache-spark_2.12-4.1.0</artifactId>
    <version>4.0.1_for_spark_3</version>
</dependency>
```

Fonctionnalités et outils Neo4j non répertoriés ici

Si vous utilisez un outil ou une fonctionnalité qui n'est pas répertorié ici, nous ne sommes pas certains de sa compatibilité avec Neptune ou les autres services qu'il propose. AWS N'hésitez pas à contacter l'équipe AWS chargée de votre compte ou à contacter l'équipe chargée de votre compte si vous avez d'autres questions.

Réécriture des requêtes Cypher pour les exécuter dans openCypher sur Neptune

Le langage openCypher est un langage de requête déclaratif pour les graphes de propriétés initialement développé par Neo4j, puis rendu open source en 2015. Il a contribué au projet openCypher sous une licence open source Apache 2. Chez AWS, nous pensons que l'open source est bénéfique pour tout le monde et nous nous engageons à apporter la valeur de l'open source à nos clients et l'excellence opérationnelle AWS aux communautés open source.

OpenCypher la syntaxe est documentée dans le Cypher Query Language Reference, version 9.

Comme openCypher contient un sous-ensemble de la syntaxe et des fonctionnalités du langage de requête Cypher, certains scénarios de migration nécessitent soit de réécrire les requêtes dans des formulaires compatibles avec openCypher, soit d'examiner d'autres méthodes pour obtenir le fonctionnement souhaité.

Cette section contient des recommandations pour gérer les différences courantes, mais elles ne sont en aucun cas exhaustives. Testez minutieusement toute application utilisant ces réécritures pour vous assurer que les résultats sont conformes à vos attentes.

Réécriture des fonctions de prédicat **`None`**, **`All`** et **`Any`**

Ces fonctions ne font pas partie de la spécification openCypher. Des résultats comparables peuvent être obtenus dans openCypher à l'aide de la compréhension de liste.

Par exemple, recherchez tous les chemins qui vont d'un nœud `Start` au nœud `End`, mais aucun trajet n'est autorisé à passer par un nœud dont la propriété de classe correspond à `D` :

```
# Neo4J Cypher code
match p=(a:Start)-[:HOP*1..]->(z:End)
where none(node IN nodes(p) where node.class ='D')
return p

# Neptune openCypher code
match p=(a:Start)-[:HOP*1..]->(z:End)
where size([node IN nodes(p) where node.class = 'D']) = 0
return p
```

La compréhension de liste peut obtenir ces résultats comme suit :

```
all  => size(list_comprehension(list)) = size(list)
```

```
any  => size(list_comprehension(list)) >= 1
none => size(list_comprehension(list)) = 0
```

Réécriture de la fonction Cypher **`reduce()`** en openCypher

La fonction `reduce()` ne fait pas partie de la spécification openCypher. Elle est souvent utilisée pour créer une agrégation de données à partir d'éléments d'une liste. Dans de nombreux cas, vous pouvez utiliser une combinaison de la compréhension de liste et de la clause `UNWIND` pour obtenir des résultats similaires.

Par exemple, la requête Cypher suivante trouve tous les aéroports situés sur des trajets comportant un à trois arrêts entre Anchorage (ANC) et Austin (AUS), et renvoie la distance totale de chaque trajet :

```
MATCH p=(a:airport {code: 'ANC'})-[r:route*1..3]->(z:airport {code: 'AUS'})
RETURN p, reduce(totalDist=0, r in relationships(p) | totalDist + r.dist) AS totalDist
ORDER BY totalDist LIMIT 5
```

Vous pouvez écrire la même requête en openCypher pour Neptune comme suit :

```
MATCH p=(a:airport {code: 'ANC'})-[r:route*1..3]->(z:airport {code: 'AUS'})
UNWIND [i in relationships(p) | i.dist] AS di
RETURN p, sum(di) AS totalDist
ORDER BY totalDist
LIMIT 5
```

Réécriture de la clause Cypher FOREACH en openCypher

La clause FOREACH ne fait pas partie de la spécification openCypher. Elle est souvent utilisée pour mettre à jour les données au milieu d'une requête, souvent à partir d'agrégations ou d'éléments au sein d'un chemin.

À titre d'exemple, recherchez tous les aéroports situés sur un trajet ne comportant pas plus de deux arrêts entre Anchorage (ANC) et Austin (AUS) et définissez la propriété « visited » pour chacun d'eux :

```
# Neo4J Example
MATCH p=(:airport {code: 'ANC'})-[*1..2]->({code: 'AUS'})
FOREACH (n IN nodes(p) | SET n.visited = true)

# Neptune openCypher
```

```
MATCH p=(:airport {code: 'ANC'})-[*1..2]->({code: 'AUS'})
WITH nodes(p) as airports
UNWIND airports as a
SET a.visited=true
```

Voici un autre exemple :

```
# Neo4J Example
MATCH p=(start)-[*]->(finish)
WHERE start.name = 'A' AND finish.name = 'D'
FOREACH (n IN nodes(p) | SET n.marked = true)

# Neptune openCypher
MATCH p=(start)-[*]->(finish)
WHERE start.name = 'A' AND finish.name = 'D'
UNWIND nodes(p) AS n
SET n.marked = true
```

Réécriture des procédures Neo4j APOC dans Neptune

Les exemples ci-dessous utilisent openCypher pour remplacer certaines des procédures APOC les plus couramment utilisées. Ces exemples sont fournis à titre de référence uniquement et visent à fournir des suggestions sur la manière de gérer les scénarios courants. Dans la pratique, chaque application est différente. Vous devrez donc élaborer vos propres stratégies pour fournir toutes les fonctionnalités dont vous avez besoin.

Réécriture des procédures **`apoc.export`**

Neptune propose un éventail d'options pour les exportations de graphes complets et les exportations basées sur des requêtes dans différents formats de sortie tels que CSV et JSON, à l'aide de l'utilitaire neptune-export (voir Exportation de données depuis un cluster de bases de données Neptune).

Réécriture des procédures **`apoc.schema`**

Neptune n'a pas de schéma, d'index ni de contraintes explicitement définis. De nombreuses procédures `apoc.schema` ne sont donc plus nécessaires. Voici quelques exemples :

- `apoc.schema.assert`
- `apoc.schema.node.constraintExists`
- `apoc.schema.node.indexExists`,
- `apoc.schema.relationship.constraintExists`

- `apoc.schema.relationship.indexExists`
- `apoc.schema.nodes`
- `apoc.schema.relationships`

Neptune openCypher prend en charge la récupération de valeurs similaires à celles des procédures, comme indiqué ci-dessous. Toutefois, il peut rencontrer des problèmes de performances sur les graphes de grande taille, car l'analyse d'une grande partie du graphe est nécessaire pour pouvoir renvoyer la réponse.

```
# openCypher replacement for apoc.schema.properties.distinct
MATCH (n:airport)
RETURN DISTINCT n.runways
```

```
# openCypher replacement for apoc.schema.properties.distinctCount
MATCH (n:airport)
RETURN DISTINCT n.runways, count(n.runways)
```

Alternatives aux procédures **`apoc.do`**

Ces procédures sont utilisées pour fournir une exécution de requête conditionnelle facile à implémenter à l'aide d'autres clauses openCypher. Dans Neptune, il existe au moins deux manières d'obtenir un comportement similaire :

- L'une des solutions consiste à combiner les fonctionnalités de compréhension de liste openCypher avec la clause `UNWIND`.
- Une autre méthode consiste à utiliser les étapes choose() et coalesce() dans Gremlin.

Des exemples de ces approches sont présentés ci-dessous.

Alternatives à apoc.do.when

```
# Neo4J Example
MATCH (n:airport {region: 'US-AK'})
CALL apoc.do.when(
 n.runways>=3,
 'SET n.is_large_airport=true RETURN n',
 'SET n.is_large_airport=false RETURN n',
 {n:n}
) YIELD value
```

```
WITH collect(value.n) as airports
RETURN size([a in airports where a.is_large_airport]) as large_airport_count,
size([a in airports where NOT a.is_large_airport]) as small_airport_count


# Neptune openCypher
MATCH (n:airport {region: 'US-AK'})
WITH n.region as region, collect(n) as airports
WITH [a IN airports where a.runways >= 3] as large_airports,
[a IN airports where a.runways < 3] as small_airports, airports
UNWIND large_airports as la
SET la.is_large_airport=true
WITH DISTINCT small_airports, airports
UNWIND small_airports as la
    SET la.small_airports=true
WITH DISTINCT airports
RETURN size([a in airports where a.is_large_airport]) as large_airport_count,
size([a in airports where NOT a.is_large_airport]) as small_airport_count

#Neptune Gremlin using choose()
g.V().
  has('airport', 'region', 'US-AK').
  choose(
    values('runways').is(lt(3)),
    property(single, 'is_large_airport', false),
    property(single, 'is_large_airport', true)).
  fold().
  project('large_airport_count', 'small_airport_count').
    by(unfold().has('is_large_airport', true).count()).
    by(unfold().has('is_large_airport', false).count())

 #Neptune Gremlin using coalesce()
g.V().
  has('airport', 'region', 'US-AK').
  coalesce(
    where(values('runways').is(lt(3))).
    property(single, 'is_large_airport', false),
    property(single, 'is_large_airport', true)).
  fold().
  project('large_airport_count', 'small_airport_count').
    by(unfold().has('is_large_airport', true).count()).
    by(unfold().has('is_large_airport', false).count())
```

Alternatives à apoc.do.case

```
# Neo4J Example
MATCH (n:airport {region: 'US-AK'})
CALL apoc.case([
 n.runways=1, 'RETURN "Has one runway" as b',
 n.runways=2, 'RETURN "Has two runways" as b'
 ],
 'RETURN "Has more than 2 runways" as b'
) YIELD value
RETURN {type: value.b,airport: n}

# Neptune openCypher
MATCH (n:airport {region: 'US-AK'})
WITH n.region as region, collect(n) as airports
WITH [a IN airports where a.runways =1] as single_runway,
[a IN airports where a.runways =2] as double_runway,
[a IN airports where a.runways >2] as many_runway
UNWIND single_runway as sr
    WITH {type: "Has one runway",airport: sr} as res, double_runway, many_runway
WITH DISTINCT double_runway as double_runway, collect(res) as res, many_runway
UNWIND double_runway as dr
    WITH {type: "Has two runways",airport: dr} as two_runways, res, many_runway
WITH collect(two_runways)+res as res, many_runway
UNWIND many_runway as mr
    WITH {type: "Has more than 2 runways",airport: mr} as res2, res, many_runway
WITH collect(res2)+res as res
UNWIND res as r
RETURN r

#Neptune Gremlin using choose()
g.V().
  has('airport', 'region', 'US-AK').
  project('type', 'airport').
    by(
      choose(values('runways')).
        option(1, constant("Has one runway")).
        option(2, constant("Has two runways")).
        option(none, constant("Has more than 2 runways"))).
    by(elementMap())

 #Neptune Gremlin using coalesce()
 g.V().
  has('airport', 'region', 'US-AK').
```

```
  project('type', 'airport').
    by(
      coalesce(
        has('runways', 1).constant("Has one runway"),
        has('runways', 2).constant("Has two runways"),
        constant("Has more than 2 runways"))).
    by(elementMap())
```

Alternatives aux propriétés basées sur des listes

Neptune ne prend actuellement pas en charge le stockage des propriétés basées sur des listes. Cependant, des résultats similaires peuvent être obtenus en stockant les valeurs de liste sous forme de chaîne séparée par des virgules, puis en utilisant les fonctions `join()` et `split()` pour construire et déconstruire la propriété de liste.

Par exemple, si nous voulions enregistrer une liste de balises en tant que propriété, nous pourrions utiliser l'exemple de réécriture qui montre comment récupérer une propriété séparée par des virgules, puis utiliser les fonctions `split()` et `join()` avec la compréhension de liste pour obtenir des résultats comparables :

```
# Neo4j Example (In this example, tags is a durable list of string.
MATCH (person:person {name: "TeeMan"})
WITH person, [tag in person.tags WHERE NOT (tag IN ['test1', 'test2', 'test3'])] AS
 newTags
SET person.tags = newTags
RETURN person

# Neptune openCypher
MATCH (person:person {name: "TeeMan"})
WITH person, [tag in split(person.tags, ',') WHERE NOT (tag IN ['test1', 'test2',
 'test3'])] AS newTags
SET person.tags = join(newTags,',')
RETURN person
```

Réécriture des sous-requêtes CALL

`CALL`Les sous-requêtes Neptune ne prennent pas en charge la syntaxe `CALL (friend) { ... }` permettant d'importer des variables dans le champ d'application de la sous-requête (dans cet `friend` exemple). Veuillez utiliser la `WITH` clause contenue dans la sous-requête pour la même chose, par exemple, `CALL { WITH friend ... }`

Les `CALL` sous-requêtes facultatives ne sont pas prises en charge pour le moment.

Autres différences entre Neptune openCypher et Cypher

- Neptune prend uniquement en charge les connexions TCP pour le protocole Bolt. WebSocketsles connexions pour Bolt ne sont pas prises en charge.
- Neptune openCypher supprime les espaces tels que définis par Unicode dans les fonctions `trim()`, `ltrim()` et `rtrim()`.
- Dans Neptune openCypher, `tostring(`double`)` ne passe pas automatiquement en notation E pour les grandes valeurs du double.

Ressources pour la migration de Neo4j vers Neptune

Neptune fournit plusieurs outils et ressources qui peuvent faciliter le processus de migration.

Outils pour faciliter la migration de Neo4j vers Neptune

- L'OpenCypher. CheatSheet
- neo4 j-to-neptune — Un utilitaire en ligne de commande pour migrer des données de Neo4j vers Neptune. Cet outil permet notamment de :
  - Exportez les données d'un graphe Neo4j correctement configuré.
  - Convertissez ces données au format Neptune.
  - Chargez ces données en masse dans Neptune.
  - Effectuez certaines conversions de base des données lors de la conversion au format Neptune, par exemple en renommant les étiquettes des sommets ou des arêtes et en générant des éléments.
  - Générez des propriétés pour les nœuds et les arêtes à l'aide de modèles (par exemple, créez une `~id` valeur à l'aide d'un modèle, comme `Person_{personid}` dans les situations où vous devez créer l'identifiant unique d'un élément).
- Vérificateur de compatibilité des requêtes OpenCypher — Cet outil prend en compte les requêtes OpenCypher et va :
  - Vérifiez la compatibilité avec la version sélectionnée de Neptune.
  - Identifiez les fonctions et clauses spécifiques non prises en charge avec leurs positions.
  - Suggérez des remplacements si disponibles.
  - Fournissez une description de toute autre erreur de syntaxe.

Migration d'un graphe existant d'un serveur Apache TinkerPop Gremlin vers Amazon Neptune

Si vous avez des données graphiques sur un serveur Apache TinkerPop Gremlin que vous souhaitez migrer vers Amazon Neptune, vous devez suivre les étapes suivantes :

1. Exportez les données du serveur Gremlin sur Amazon Simple Storage Service (Amazon S3).
2. Convertissez les données exportées dans un format CSV dans lequel le chargeur en bloc Neptune peut importer les données.
3. À l'aide du Chargeur en bloc Neptune, importez les données dans un cluster de bases de données Neptune que vous avez préparé.
4. Modifiez votre application existante pour vous connecter au point de terminaison Gremlin de Neptune et apportez les modifications nécessaires pour vous conformer aux différences d'implémentation de Neptune Gremlin.

Migration d'un graphe existant d'un triplestore RDF vers Amazon Neptune

Si vous avez des données graphiques RDF/SPARQL à migrer vers Amazon Neptune, vous devez suivre les étapes suivantes :

1. Exportez les données de votre triplestore RDF.
2. Convertissez les données exportées dans un format dans lequel le chargeur en bloc Neptune peut importer les données.
3. Stockez les données à importer dans Amazon Simple Storage Service (Amazon S3).
4. À l'aide du Chargeur en bloc Neptune, importez les données d'Amazon S3 dans un cluster de bases de données Neptune que vous avez préparé.
5. Modifiez votre application existante pour vous connecter au point de terminaison SPARQL de Neptune.

Si vous souhaitez essayer de migrer des données CSV de graphes de propriétés vers RDF, vous pouvez utiliser le convertisseur de CSV en RDF Amazon Neptune.

Utilisation de AWS Database Migration Service (AWS DMS) pour migrer d'une base de données relationnelle ou NoSQL vers Amazon Neptune

AWS Database Migration Service (AWS DMS) est un service cloud qui facilite la migration de bases de données relationnelles, d'entrepôts de données, de bases de données NoSQL et d'autres types de magasins de données. Si vous avez des données graphiques stockées dans l'une des bases de données relationnelles ou NoSQL compatibles AWS DMS , AWS DMS cela peut vous aider à migrer vers Neptune rapidement et en toute sécurité, sans que votre base de données actuelle ne soit interrompue. Consultez Utilisation AWS Database Migration Service pour charger des données dans Amazon Neptune à partir d'un autre magasin de données pour plus de détails.

Le flux de données de migration utilisé AWS DMS est le suivant :

- Créez un objet de AWS DMS mappage de table. Cet objet JSON spécifie les tables qui doivent être lues à partir de la base de données source, l'ordre dans lequel la lecture doit s'effectuer, ainsi que le nom des colonnes. Il peut également filtrer les lignes copiées et fournir des transformations de valeur simples telles que la conversion en minuscules ou l'arrondi.
- Créez une configuration `GraphMappingConfig` Neptune pour spécifier la façon dont les données extraites de la base de données source doivent être chargées dans Neptune.
  - Pour les données RDF (interrogées à l'aide de SPARQL), l'élément `GraphMappingConfig` est écrit dans le langage de mappage standard W3 R2RML.
  - Pour les données de graphe de propriétés (interrogées à l'aide de Gremlin), `GraphMappingConfig` est un objet JSON, comme décrit dans GraphMappingConfig Mise en page des Property-Graph/Gremlin données.
- Créez une instance de AWS DMS réplication dans le même VPC que votre cluster de base de données Neptune pour effectuer la migration.
- Créez un compartiment Amazon S3 à utiliser comme stockage intermédiaire pour préparer les données en cours de migration.
- Exécutez la tâche de AWS DMS migration.

Pour plus d'informations, consultez le billet de blog en quatre parties de Chris Smith, « Populating your graph in Amazon Neptune from a relational database using AWS Database Migration Service (DMS) » :

- Partie 1 : Préparer le terrain
- Partie 2 : Conception du modèle de graphe de propriétés
- Partie 3 : Conception du modèle RDF
- Partie 4 : synthèse

Migration de Blazegraph vers Amazon Neptune

Si vous avez un graphe dans le triplestore RDF open source Blazegraph, vous pouvez migrer ses données vers Amazon Neptune en suivant les étapes ci-dessous :

- AWS Infrastructure de fourniture. Commencez par provisionner l'infrastructure Neptune requise à l'aide d' AWS CloudFormation un modèle (voir)Créer un cluster Neptune.
- Exportez les données à partir de Blazegraph. Il existe deux méthodes principales pour exporter des données à partir de Blazegraph : les requêtes SPARQL CONSTRUCT ou l'utilitaire d'exportation Blazegraph.
- Importez les données dans Neptune. Vous pouvez ensuite charger les fichiers de données exportés dans Neptune à l'aide du workbench Neptune et de Chargeur en bloc Neptune.

En règle générale, cette approche s'applique également à la migration à partir d'autres bases de données triplestore RDF.

Compatibilité entre Blazegraph et Neptune

Avant de migrer les données de votre graphe vers Neptune, il est important de connaître plusieurs différences importantes entre Blazegraph et Neptune. Ces différences peuvent nécessiter des modifications des requêtes, de l'architecture de l'application ou des deux. Elles peuvent même compliquer la migration :

- **`Full-text search`** : dans Blazegraph, vous pouvez utiliser des fonctionnalités de recherche en texte intégral internes ou externes grâce à une intégration avec Apache Solr. Le cas échéant, informez-vous des dernières mises à jour concernant les fonctionnalités de recherche en texte intégral prises en charge par Neptune. Consultez Recherche en texte intégral Neptune.
- **`Query hints`** : Blazegraph et Neptune étendent SPARQL avec le concept d'indicateurs de requête. Au cours d'une migration, vous devez migrer tous les indicateurs de requête que vous utilisez. Pour en savoir plus sur les derniers indicateurs de requête pris en charge par Neptune, consultez Indicateurs de requête SPARQL.
- Inférence : Blazegraph prend en charge l'inférence en tant qu'option configurable en mode triplet, mais pas en mode quadruplet. Neptune ne prend pas encore en charge l'inférence.
- Recherche géospatiale : Blazegraph prend en charge la configuration d'espaces de noms qui permettent la prise en charge géospatiale. Cette fonctionnalité n'est pas encore disponible dans Neptune.

- Architecture mutualisée : Blazegraph prend en charge l'architecture mutualisée au sein d'une base de données unique. Dans Neptune, l'architecture mutualisée est prise en charge soit en stockant les données dans des graphes nommés et en utilisant les clauses USING NAMED pour les requêtes SPARQL, soit en créant un cluster de bases de données distinct pour chaque locataire.
- Fédération — Neptune prend actuellement en charge la fédération SPARQL 1.1 vers des emplacements accessibles à l'instance Neptune, par exemple au sein du VPC privé, via ou vers des points de terminaison Internet externes. VPCs En fonction de la configuration spécifique et des points de terminaison de fédération requis, vous aurez peut-être besoin d'une configuration réseau supplémentaire.
- Extensions des normes Blazegraph : Blazegraph inclut plusieurs extensions aux normes d'API REST et SPARQL, tandis que Neptune n'est compatible qu'avec les spécifications de normes elles-mêmes. Cela peut nécessiter des modifications de votre application ou rendre la migration difficile.

AWS Infrastructure de provisionnement pour Neptune

Bien que vous puissiez créer l' AWS infrastructure requise manuellement via le AWS Management Console ou AWS CLI, il est souvent plus pratique d'utiliser un CloudFormation modèle, comme décrit ci-dessous :

Approvisionnement de Neptune à l'aide CloudFormation d'un modèle :

1. Accédez à Création d'un cluster Amazon Neptune à l'aide de AWS CloudFormation.
2. Choisissez Launcher la pile dans votre région préférée.
3. Définissez les paramètres obligatoires (nom de la pile et `EC2SSHKeyPairName`). Définissez également les paramètres facultatifs suivants pour faciliter le processus de migration :
   - Définissez `AttachBulkloadIAMRoleToNeptuneCluster` sur true. Ce paramètre permet de créer et d'associer le rôle IAM approprié à votre cluster afin de permettre le chargement en bloc des données.
   - Définissez `NotebookInstanceType` sur le type d'instance que vous préférez. Ce paramètre crée un classeur Neptune que vous utiliserez pour exécuter le chargement en bloc dans Neptune et pour valider la migration.
4. Choisissez Suivant.
5. Définissez toutes les autres options que vous souhaitez pour la pile.
6. Choisissez Suivant.

7. Passez en revue vos options et cochez les deux cases pour reconnaître que des fonctionnalités supplémentaires AWS CloudFormation peuvent être nécessaires.
8. Sélectionnez Créer la pile.

Le processus de création de la pile peut prendre quelques minutes.

Exportation de données à partir de Blazegraph

L'étape suivante consiste à exporter les données à partir de Blazegraph dans un format compatible avec le chargeur en bloc Neptune.

En fonction de la manière dont les données sont stockées dans Blazegraph (triplets ou quadruplets) et du nombre de graphes nommés utilisés, Blazegraph peut vous demander d'effectuer le processus d'exportation plusieurs fois et de générer plusieurs fichiers de données :

- Si les données sont stockées sous forme de triplets, vous devez exécuter une exportation pour chaque graphe nommé.
- Si les données sont stockées sous forme de quadruplets, vous pouvez choisir d'exporter les données au format N-Quads ou d'exporter chaque graphe nommé sous forme de triplets.

Ci-dessous, nous supposons que vous exportez un seul espace de noms sous forme de N-Quads, mais vous pouvez répéter le processus pour des espaces de noms supplémentaires ou pour les formats d'exportation souhaités.

Si Blazegraph doit être en ligne et disponible pendant la migration, utilisez des requêtes SPARQL CONSTRUCT. Cela nécessite que vous installiez, configuriez et exécutiez une instance Blazegraph avec un point de terminaison SPARQL accessible.

Si vous n'avez pas besoin de Blazegraph pour être en ligne, utilisez l'utilitaire BlazeGraph d'exportation. Pour ce faire, vous devez télécharger Blazegraph. Le fichier de données et les fichiers de configuration doivent être accessibles, mais le serveur n'a pas besoin d'être en cours d'exécution.

Exportation de données à partir de Blazegraph à l'aide de SPARQL CONSTRUCT

SPARQL CONSTRUCT est une fonctionnalité SPARQL qui renvoie un graphe RDF correspondant au modèle de requête spécifié. Dans ce cas d'utilisation, vous pouvez l'utiliser pour exporter les données un espace de noms à la fois, à l'aide d'une requête comme celle-ci :

```
CONSTRUCT WHERE { hint:Query hint:analytic "true" . hint:Query
 hint:constructDistinctSPO "false" . ?s ?p ?o }
```

Bien que d'autres outils RDF permettent d'exporter ces données, le moyen le plus simple d'exécuter cette requête est d'utiliser le point de terminaison de l'API REST, fourni par Blazegraph. Le script suivant montre comment utiliser un script Python (version 3.6 ou ultérieure) pour exporter des données au format N-Quads :

```
import requests

# Configure the URL here: e.g. http://localhost:9999/sparql
url = "http://localhost:9999/sparql"
payload = {'query': 'CONSTRUCT WHERE { hint:Query hint:analytic "true" . hint:Query
 hint:constructDistinctSPO "false" . ?s ?p ?o }'}
# Set the export format to be n-quads
headers = {
'Accept': 'text/x-nquads'
}
# Run the http request
response = requests.request("POST", url, headers=headers, data = payload, files = [])
#open the file in write mode, write the results, and close the file handler
f = open("export.nq", "w")
f.write(response.text)
f.close()
```

Si les données sont stockées sous forme de triples, vous devez modifier le paramètre d'`Accept`en-tête pour exporter les données dans un format approprié (N-Triples, RDF/XML ou Turtle) en utilisant les valeurs spécifiées dans le dépôt Blazegraph. GitHub

Exportation de données à l'aide de l'utilitaire d'exportation Blazegraph

Blazegraph contient une méthode d'utilitaire pour exporter les données, à savoir la classe `ExportKB`. `ExportKB` facilite l'exportation des données à partir de Blazegraph, mais contrairement à la méthode précédente, nécessite que le serveur soit hors ligne pendant l'exportation. Il s'agit de la méthode de choix lorsque vous pouvez mettre Blazegraph hors ligne pendant la migration ou lorsque la migration peut se faire à partir d'une sauvegarde des données.

Vous devez exécuter l'utilitaire à partir d'une ligne de commande Java sur un ordinateur sur lequel Blazegraph est installé, mais ne fonctionne pas. Le moyen le plus simple d'exécuter cette commande

est de télécharger la dernière version de blazegraph.jar située sur GitHub. L'exécution de cette commande nécessite plusieurs paramètres :

- **`log4j.primary.configuration`** : emplacement du fichier de propriétés log4j.
- **`log4j.configuration`** : emplacement du fichier de propriétés log4j.
- **`output`** : répertoire de sortie des données exportées. Les fichiers au format `tar.gz` se trouvent dans un sous-répertoire nommé comme indiqué dans la base de connaissances.
- **`format`** : format de sortie souhaité suivi de l'emplacement du fichier `RWStore.properties`. Si vous travaillez avec des triplets, vous devez remplacer le paramètre `-format` par `N-Triples`, `Turtle` ou `RDF/XML`.

Par exemple, si vous disposez du fichier journal et des fichiers de propriétés Blazegraph, exportez les données au format N-Quads à l'aide du code suivant :

```
java -cp blazegraph.jar \
    com.bigdata.rdf.sail.ExportKB \
    -outdir ~/temp/ \
    -format N-Quads \
    ./RWStore.properties
```

Si l'exportation réussit, la sortie ressemble à ce qui suit :

```
Exporting kb as N-Quads on /home/ec2-user/temp/kb
Effective output directory: /home/ec2-user/temp/kb
Writing /home/ec2-user/temp/kb/kb.properties
Writing /home/ec2-user/temp/kb/data.nq.gz
Done
```

Création d'un compartiment Amazon Simple Storage Service (Amazon S3) pour y copier les données exportées

Une fois que vous avez exporté vos données à partir de Blazegraph, créez un compartiment Amazon Simple Storage Service (Amazon S3) dans la même région que le cluster de bases de données Neptune cible pour que le chargeur en bloc Neptune puisse l'utiliser pour importer les données.

Pour obtenir des instructions sur la création d'un compartiment Amazon S3, consultez Comment créer un compartiment S3 ? dans le Guide de l'utilisateur Amazon Simple Storage Service, ainsi que

des exemples de création d'un compartiment dans le Guide de l'utilisateur Amazon Simple Storage Service.

Pour savoir comment copier les fichiers de données que vous avez exportés dans le nouveau compartiment Amazon S3, consultez la section Chargement d'un objet vers un compartiment dans le guide de l'utilisateur d'Amazon Simple Storage Service ou Utilisation de commandes de haut niveau (s3) avec la AWS CLI. Vous pouvez également utiliser le code Python suivant pour copier les fichiers un par un :

```
import boto3

region = 'region name'
bucket_name = 'bucket name'
s3 = boto3.resource('s3')
s3.meta.client.upload_file('export.nq', bucket_name, 'export.nq')
```

Utilisation du chargeur en bloc Neptune pour importer les données dans Neptune

Après avoir exporté vos données à partir de Blazegraph et après les avoir copiées dans un compartiment Amazon S3, vous pouvez les importer dans Neptune. Neptune dispose d'un chargeur en bloc qui charge les données plus rapidement et à moindre coût par rapport aux opérations de chargement effectuées à l'aide de SPARQL. Le processus de chargement en bloc est lancé par un appel à l'API du point de terminaison du chargeur afin de charger dans Neptune les données stockées dans le compartiment S3 identifié.

Bien que vous puissiez effectuer un appel direct au point de terminaison REST du chargeur, vous devez avoir accès au VPC privé dans lequel s'exécute l'instance Neptune cible. Vous pourriez configurer un hôte bastion, l'associer avec une connexion SSH à cet ordinateur et exécuter la commande cURL, mais l'utilisation du workbench Neptune est plus simple.

Neptune Workbench est un bloc-notes Jupyter préconfiguré fonctionnant comme un bloc-notes Amazon, sur lequel sont installés plusieurs SageMaker blocs-notes spécifiques à Neptune. Ces magies simplifient les opérations Neptune courantes, telles que la vérification du statut du cluster, l'exécution de traversées SPARQL et Gremlin, ainsi que l'exécution d'une opération de chargement en bloc.

Pour démarrer le processus de chargement en bloc, utilisez la magie `%load`, qui fournit une interface permettant d'exécuter la Commande de chargeur Neptune :

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Sélectionnez aws-neptune-blazegraph-to-neptune.
3. Choisissez Ouvrir un bloc-notes.
4. Dans l'instance en cours d'exécution de Jupyter, sélectionnez un bloc-notes existant ou créez-en un à l'aide du noyau Python 3.
5. Dans le bloc-notes, ouvrez une cellule, entrez `%load`, puis exécutez-la.
6. Définissez les paramètres du chargeur en bloc :
   a. Pour Source, entrez l'emplacement du fichier source à importer : `s3://{bucket_name}/{file_name}`.
   b. Pour Format, choisissez le format approprié, qui est `nquads` dans cet exemple.
   c. Pour ARN du chargement, entrez l'ARN du rôle `IAMBulkLoad` (ces informations se trouvent dans la console IAM sous Rôles).
7. Sélectionnez Envoyer.

Le résultat contient le statut de la demande. Les chargements en bloc sont souvent des processus de longue durée. La réponse ne signifie donc pas que le chargement est terminé, mais seulement qu'il a commencé. Ces informations de statut sont mises à jour périodiquement jusqu'à ce qu'elles indiquent que la tâche est terminée.

> **Note**
>
> Ces informations sont également disponibles dans le billet de blog Moving to the cloud : Migrating Blazegraph to Amazon Neptune.

Chargement de données dans Amazon Neptune

Il existe plusieurs façons de charger des données de graphe dans Amazon Neptune :

- Si vous n'avez besoin de charger qu'une quantité relativement faible de données, vous pouvez utiliser des requêtes telles que les `INSERT` instructions SPARQL ou Gremlin `mergeV` et `mergeE` les étapes. OpenCypher inclut également des clauses and. `CREATE` `MERGE`
- Vous pouvez profiter des avantages offerts par Chargeur en bloc Neptune pour ingérer de grandes quantités de données résidant dans des fichiers externes. La commande de chargement en bloc est plus rapide et demande moins de ressources que les commandes de langage de requête. Elle est optimisée pour les jeux de données volumineux et prend en charge les données RDF (Resource Description Framework) et les données Gremlin.
- Vous pouvez utiliser AWS Database Migration Service (AWS DMS) pour importer des données depuis d'autres magasins de données (voir Utilisation AWS Database Migration Service pour charger des données dans Amazon Neptune à partir d'un autre magasin de données et Guide de AWS Database Migration Service l'utilisateur).
- Pour les petits ensembles de données contenus dans un ou plusieurs fichiers Amazon S3, vous pouvez utiliser des fonctions de chargement basées sur des requêtes pour lire et traiter les données directement dans vos requêtes. Pour plus d'informations, consultez Chargement de données dans Amazon Neptune à l'aide de requêtes.

Rubriques

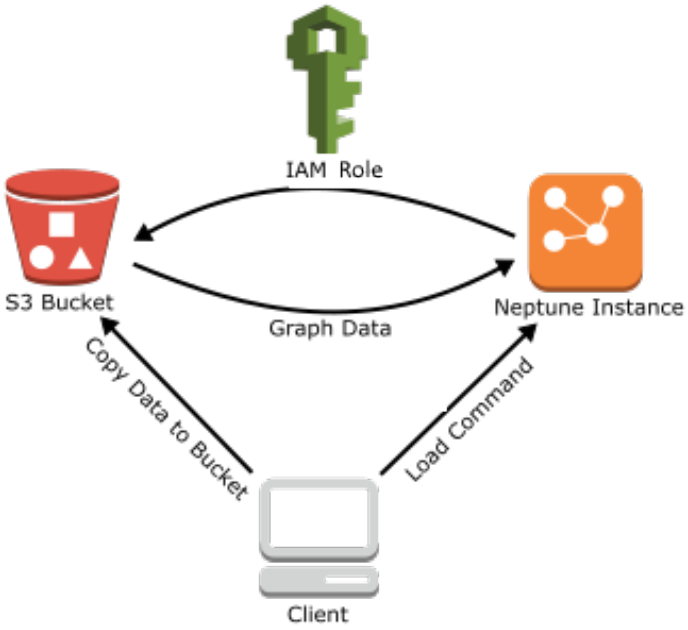
- Utilisation du chargeur en vrac Amazon Neptune pour ingérer des données
- Utilisation AWS Database Migration Service pour charger des données dans Amazon Neptune à partir d'un autre magasin de données
- Chargement de données dans Amazon Neptune à l'aide de requêtes

Utilisation du chargeur en vrac Amazon Neptune pour ingérer des données

Amazon Neptune fournit une commande `Loader` pour le chargement de données depuis des fichiers externes directement dans un cluster de bases de données Neptune. Vous pouvez utiliser cette commande au lieu d'exécuter un grand nombre de déclarations `INSERT`, d'étapes `addV` et `addE` ou d'autres appels d'API.

La commande Neptune Loader est plus rapide, nécessite moins de ressources, est optimisée pour les jeux de données volumineux et prend en charge à la fois les données RDF (Resource Description Framework) et Gremlin utilisées par SPARQL.

Le diagramme suivant présente un aperçu du processus de chargement.



Voici les étapes du processus de chargement :

1. Chargez le fichier de données dans un compartiment Amazon Simple Storage Service (Amazon S3).
2. Créer un rôle IAM avec un accès en lecture et de type Liste au compartiment.
3. Créez un point de terminaison de VPC Amazon S3
4. Pour démarrer le chargeur Neptune, envoyez une demande via HTTP à l'instance de base de données Neptune.
5. L'instance de base de données Neptune endosse le rôle IAM pour charger les données à partir du compartiment.

> **Note**
>
> Vous pouvez charger des données chiffrées depuis Amazon S3 si elles ont été chiffrées à l'aide d'Amazon S3 `SSE-S3` ou du `SSE-KMS` mode, à condition que le rôle que vous utilisez pour le chargement en masse ait accès à l'objet Amazon S3 et, dans le cas de SSE-KMS,

àkms:decrypt. Neptune pourra ainsi emprunter vos informations d'identification et émettre des appels s3:getObject en votre nom.
Cependant, à ce stade, Neptune ne permet pas lle chargement de données chiffrées avec le mode SSE-C.

Les sections suivantes fournissent des instructions pour la préparation et le chargement des données dans Neptune.

Rubriques

- Prérequis : rôle IAM et accès à Amazon S3
- Formats de chargement de données
- Exemple : chargement de données dans une instance de base de données Neptune
- Optimisation d'un chargement en bloc sur Amazon Neptune
- Référence du chargeur Neptune

Prérequis : rôle IAM et accès à Amazon S3

Le chargement de données depuis un compartiment Amazon Simple Storage Service (Amazon S3) nécessite Gestion des identités et des accès AWS un rôle (IAM) ayant accès au compartiment. Amazon Neptune endossera ce rôle pour charger les données.

Note

Vous pouvez charger des données chiffrées à partir d'Amazon S3 si elles ont été chiffrées avec le mode SSE-S3 d'Amazon S3. Dans ce cas, Neptune peut emprunter vos informations d'identification et émettre des appels s3:getObject en votre nom.
Vous pouvez également charger les données chiffrées à l'aide du mode SSE-KMS à partir d'Amazon S3, à condition que votre rôle IAM implique les autorisations nécessaires pour accéder à AWS KMS. Sans AWS KMS autorisations appropriées, l'opération de chargement en bloc échoue et renvoie une LOAD_FAILED réponse.
Neptune ne permet actuellement pas le chargement des données chiffrées Amazon S3 avec le mode SSE-C.

Les sections suivantes montrent comment utiliser une politique IAM gérée pour créer un rôle IAM permettant d'accéder aux ressources Amazon S3, puis attacher le rôle à votre cluster Neptune.

Rubriques

- Création d'un rôle IAM pour autoriser Amazon Neptune à accéder aux ressources Amazon S3
- Ajout du rôle IAM à un cluster Amazon Neptune
- Création du point de terminaison de VPC Amazon S3
- Chaînage des rôles IAM dans Amazon Neptune

> **Note**
>
> Pour ces instructions, vous devez disposer d'un accès à la console IAM et des autorisations nécessaires pour gérer les rôles et les politiques IAM. Pour plus d'informations, consultez la section Autorisations de travail dans la console AWS de gestion dans le guide de l'utilisateur IAM.
> La console Amazon Neptune a besoin que l'utilisateur dispose des autorisations IAM suivantes pour attacher le rôle au cluster Neptune :
>
> ```
> iam:GetAccountSummary on resource: *
> iam:ListAccountAliases on resource: *
> iam:PassRole on resource: * with iam:PassedToService restricted to
>  rds.amazonaws.com
> ```

Création d'un rôle IAM pour autoriser Amazon Neptune à accéder aux ressources Amazon S3

Utilisez la politique IAM gérée `AmazonS3ReadOnlyAccess` pour créer un rôle IAM qui permet à Amazon Neptune d'accéder aux ressources Amazon S3.

Pour créer un rôle IAM autorisant Neptune à accéder à Amazon S3

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le panneau de navigation, choisissez Rôles.
3. Sélectionnez Create role (Créer un rôle).
4. Sous Service AWS , sélectionnez S3.

5. Choisissez Suivant : Autorisations.
6. Utilisez la case de filtre pour filtrer par le terme S3 et cochez la case à côté de AmazonS3ReadOnlyAccess.

> **Note**
>
> Cette stratégie accorde les autorisations `s3:Get*` et `s3:List*` à tous les compartiments. Les étapes ultérieures limitent l'accès au rôle à l'aide de la stratégie d'approbation.
> Le chargeur requiert uniquement les autorisations `s3:Get*` et `s3:List*` au niveau du compartiment à partir duquel vous effectuez le chargement. Vous pouvez donc également restreindre ces autorisations par la ressource Amazon S3.
> Si votre compartiment S3 est chiffré, vous devez ajouter les autorisations `kms:Decrypt`

7. Choisissez Suivant : Vérification.
8. Dans Nom du rôle, attribuez un nom au rôle IAM (par exemple, `NeptuneLoadFromS3`). Vous pouvez également ajouter une valeur facultative dans le champ Description du rôl) (par exemple, « Autorise Neptune à accéder aux ressources S3 en votre nom »).
9. Choisissez Create Role (Créer un rôle).
10. Dans le panneau de navigation, choisissez Rôles.
11. Dans le champ Search (Rechercher), saisissez le nom du rôle que vous avez créé, puis choisissez celui-ci quand il s'affiche dans la liste.
12. Dans l'onglet Trust Relationships (Relations d'approbation), choisissez Edit trust relationship (Modifier la relation d'approbation).
13. Dans la zone de texte, copiez la stratégie d'approbation suivante.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "rds.amazonaws.com"
```

```
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

14. Choisissez Update Trust Policy (Mettre à jour la stratégie d'approbation).
15. Suivez les étapes de Ajout du rôle IAM à un cluster Amazon Neptune.

Ajout du rôle IAM à un cluster Amazon Neptune

Utilisez la console pour ajouter le rôle IAM à un cluster Amazon Neptune. Cela permet à toute instance de base de données Neptune du cluster d'endosser le rôle et d'effectuer un chargement depuis Amazon S3.

> **Note**
>
> La console Amazon Neptune a besoin que l'utilisateur dispose des autorisations IAM suivantes pour attacher le rôle au cluster Neptune :
>
> ```
> iam:GetAccountSummary on resource: *
> iam:ListAccountAliases on resource: *
> iam:PassRole on resource: * with iam:PassedToService restricted to
>  rds.amazonaws.com
> ```

Pour ajouter un rôle IAM à un cluster Amazon Neptune

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez l'identifiant du cluster que vous souhaitez modifier.
4. Choisissez l'onglet Connectivité et sécurité.
5. Dans la section Rôles IAM, choisissez le rôle que vous avez créé dans la section précédente.
6. Choisissez Ajouter un rôle.
7. Attendez que le rôle IAM devienne accessible au cluster avant de l'utiliser.

Création du point de terminaison de VPC Amazon S3

Le chargeur Neptune a besoin d'un point de terminaison de VPC de type passerelle pour Amazon S3.

Pour configurer l'accès à Amazon S3

1. Connectez-vous à la console Amazon VPC AWS Management Console et ouvrez-la à l'adresse. https://console.aws.amazon.com/vpc/
2. Dans le panneau de navigation, choisissez Points de terminaison.
3. Choisissez Create Endpoint (Créer un point de terminaison).
4. Choisissez le nom de service `com.amazonaws.region.s3` pour le point de terminaison de type passerelle.

   > **Note**
   >
   > Si la région dans ce champ est incorrecte, assurez-vous que la région de la console est correcte.

5. Choisissez le VPC qui contient l'instance de base de données Neptune (vous le trouverez dans la console Neptune).
6. Cochez la case en regard des tables de routage associées aux sous-réseaux liés à votre cluster. Si vous n'avez qu'une seule table de routage, vous devez cocher cette case.
7. Choisissez Créer un point de terminaison.

Pour plus d'informations sur la création du point de terminaison, consultez Points de terminaison de VPC dans le Guide de l'utilisateur Amazon VPC. Pour plus d'informations sur les limites qui s'appliquent aux points de terminaison d'un VPC, consultez Points de terminaisons de VPC pour Amazon S3.

Étapes suivantes

Maintenant que vous avez accordé l'accès au compartiment Amazon S3, vous pouvez préparer le chargement des données. Pour plus d'informations sur les formats pris en charge, consultez Formats de chargement de données.

Chaînage des rôles IAM dans Amazon Neptune

> **Important**
>
> La nouvelle fonctionnalité de chargement en bloc entre comptes, introduite dans la version 1.2.1.0.R3 du moteur, tire parti du chaînage des rôles IAM et peut dans certains cas entraîner une dégradation des performances de chargement en bloc. Par conséquent, les mises à niveau vers les versions du moteur prenant en charge cette fonctionnalité ont été temporairement suspendues jusqu'à ce que le problème soit résolu.

Lorsque vous attachez un rôle à un cluster, ce dernier peut endosser ce rôle afin d'accéder aux données stockées dans Amazon S3. À partir de la version 1.2.1.0.R3 du moteur, si ce rôle n'a pas accès à toutes les ressources dont vous avez besoin, vous pouvez enchaîner un ou plusieurs rôles supplémentaires que votre cluster endossera pour accéder à d'autres ressources. Chaque rôle de la chaîne passe au rôle suivant, jusqu'à ce que le cluster endosse le dernier rôle à la fin de la chaîne.

Pour enchaîner des rôles, vous devez établir une relation d'approbation entre eux. Par exemple, pour enchaîner `RoleB` à `RoleA`, `RoleA` doit disposer d'une politique d'autorisations lui permettant d'endosser `RoleB`, et `RoleB` d'une politique d'approbation lui permettant de retransmettre ses autorisations à `RoleA`. Pour plus d'informations, consultez Utilisation de rôles IAM.

Le premier rôle de la chaîne doit être attaché au cluster qui charge les données.

Le premier rôle, ainsi que chaque rôle ultérieur qui endosse le rôle suivant dans la chaîne, doivent avoir les éléments suivants :

- Une politique qui inclut une déclaration spécifique ayant un effet `Allow` sur l'action `sts:AssumeRole`.
- L'Amazon Resource Name (ARN) du rôle dans un élément `Resource`.

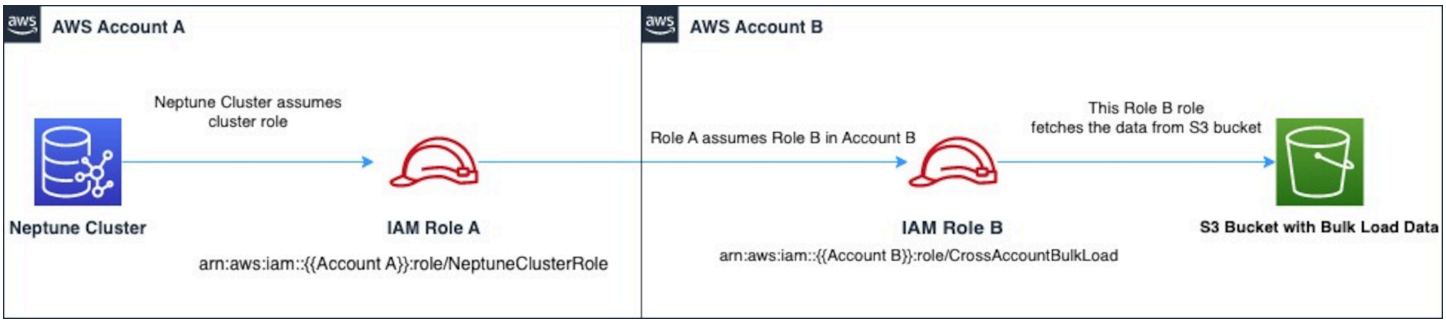
> **Note**
>
> Le compartiment Amazon S3 cible doit se trouver dans la même AWS région que le cluster.

Cross-account accès à l'aide de rôles enchaînés

Vous pouvez accorder un accès intercompte en enchaînant un ou plusieurs rôles appartenant à un autre compte. Lorsque votre cluster endosse temporairement un rôle appartenant à un autre compte, il peut accéder aux ressources qui s'y trouvent.

Supposons, par exemple, que le compte A souhaite accéder à des données dans un compartiment Amazon S3 appartenant au compte B :

- Le compte A crée un rôle de AWS service nommé pour Neptune `RoleA` et l'attache à un cluster.
- Le compte B crée un rôle nommé `RoleB` qui est autorisé à accéder aux données du compartiment du compte B.
- Le compte A attache une politique d'autorisation à `RoleA` qui lui permet d'endosser `RoleB`.
- Le compte B attache une politique d'approbation à `RoleB` qui lui permet de retransmettre ses autorisations à `RoleA`.
- Pour accéder aux données dans le compartiment du compte B, le compte A exécute une commande de chargeur à l'aide d'un paramètre `iamRoleArn` qui enchaîne les rôles `RoleA` et `RoleB`. Pendant la durée de l'opération de chargeur, `RoleA` endosse temporairement `RoleB` pour accéder au compartiment Amazon S3 dans le compte B.



Par exemple, `RoleA` a une politique d'approbation établissant une relation d'approbation avec Neptune :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
          "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

`RoleA` a également une politique d'autorisation lui permettant d'endosser `RoleB`, qui appartient au compte B :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1487639602000",
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/RoleB"
        }
    ]
}
```

Inversement, `RoleB` a une politique d'approbation établissant une relation d'approbation avec `RoleA` :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {
```

```
                "AWS": "arn:aws:iam::111122223333:role/RoleA"
            }
        }
    ]
}
```

`RoleB` a également besoin d'une autorisation pour accéder aux données du compartiment Amazon S3 situé dans le compte B.

Création d'un AWS Security Token Service Point de terminaison VPC (STS)

Le chargeur Neptune nécessite un point de terminaison VPC AWS STS lorsque vous chaînez des rôles IAM à des API d'accès AWS STS privé via des adresses IP privées. Vous pouvez vous connecter directement d'un Amazon VPC à un point AWS STS de terminaison VPC de manière sécurisée et évolutive. Lorsque vous utilisez un point de terminaison de VPC d'interface, votre niveau de sécurité s'en trouve amélioré, car vous n'avez pas besoin d'ouvrir de pare-feu pour le trafic sortant. Vous bénéficiez en outre d'autres avantages liés à l'utilisation des points de terminaison Amazon VPC.

Lorsque vous utilisez un point de terminaison VPC, le trafic vers AWS STS n'est pas transmis via Internet et ne quitte jamais le réseau Amazon. Votre VPC est connecté en toute sécurité, AWS STS sans risques de disponibilité ni contraintes de bande passante pour le trafic réseau. Pour plus d'informations, consultez Utilisation des points de terminaison de VPC d'interface AWS STS.

Pour configurer l'accès pour AWS Security Token Service (SETS)

1. Connectez-vous à la console Amazon VPC AWS Management Console et ouvrez-la à l'adresse. https://console.aws.amazon.com/vpc/
2. Dans le panneau de navigation, choisissez Points de terminaison.
3. Choisissez Create Endpoint (Créer un point de terminaison).
4. Choisissez le nom de service : `com.amazonaws.region.sts` pour le point de terminaison de type interface.
5. Choisissez le VPC qui contient votre instance de base de données Neptune et votre instance EC2.
6. Cochez la case située à côté du sous-réseau dans lequel se trouve votre instance EC2. Il n'est pas possible de sélectionner plusieurs sous-réseaux dans la même zone de disponibilité.
7. Pour IP address type (Type d'adresse IP), choisissez l'une des options suivantes :

- IPv4 – Attribuez des adresses IPv4 aux interfaces réseau de vos points de terminaison. Cette option n'est prise en charge que si tous les sous-réseaux sélectionnés possèdent des plages d'adresses IPv4.
- IPv6 – Attribuez des adresses IPv6 aux interfaces réseau de vos points de terminaison. Cette option n'est prise en charge que si tous les sous-réseaux sélectionnés sont des IPv6-only sous-réseaux.
- Dualstack – Attribuez à la fois des adresses IPv4 et IPv6 aux interfaces réseau de vos points de terminaison. Cette option n'est prise en charge que si tous les sous-réseaux sélectionnés possèdent des plages d'adresses IPv4 et IPv6.

8. Pour Security groups (Groupes de sécurité), sélectionnez les groupes de sécurité à associer aux interfaces réseau du point de terminaison pour le point de terminaison d'un VPC. Vous devez sélectionner tous les groupes de sécurité attachés à votre instance de base de données Neptune et à votre instance EC2.
9. Pour Policy (Politique), sélectionnez Full access (Accès complet) pour autoriser toutes les opérations de tous les principaux sur toutes les ressources via le point de terminaison de VPC. Sinon, sélectionnez Custom (Personnalisé) pour joindre une politique de point de terminaison de VPC qui contrôle les autorisations dont disposent les principaux pour effectuer des actions sur les ressources via le point de terminaison de VPC. Cette option n'est disponible que si le service prend en charge les politiques de points de terminaison de VPC. Pour plus d'informations, consultez Utilisation des politiques de point de terminaison.
10. (Facultatif) Pour ajouter une balise, choisissez Ajouter une nouvelle balise et saisissez la clé et la valeur de cette balise.
11. Choisissez Créer un point de terminaison.

Pour plus d'informations sur la création du point de terminaison, consultez Points de terminaison de VPC dans le Guide de l'utilisateur Amazon VPC. Notez que le point de terminaison de VPC Amazon STS est un prérequis pour le chaînage des rôles IAM.

Maintenant que vous avez accordé l'accès au AWS STS point de terminaison, vous pouvez vous préparer à charger les données. Pour en savoir plus sur les formats de données pris en charge, consultez Formats de chargement de données.

Chaînage des rôles dans une commande de chargeur

Vous pouvez spécifier le chaînage des rôles lorsque vous exécutez une commande de chargeur en incluant une liste d'ARN de rôles, séparés par des virgules, dans le paramètre `iamRoleArn`.

Bien que vous n'ayez généralement besoin que de deux rôles dans une chaîne, il est possible d'en enchaîner trois ou plus. Par exemple, cette commande de chargeur enchaîne trois rôles :

AWS CLI

```
aws neptunedata start-loader-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --source "s3://(the target bucket name)/(the target date file name)" \
  --format "csv" \
  --iam-role-arn "arn:aws:iam::(Account A ID):role/(RoleA),arn:aws:iam::(Account B
 ID):role/(RoleB),arn:aws:iam::(Account C ID):role/(RoleC)" \
  --s3-bucket-region "us-east-1"
```

Pour plus d'informations, consultez start-loader-job dans la AWS CLI référence des commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_loader_job(
    source='s3://(the target bucket name)/(the target date file name)',
    format='csv',
    iamRoleArn='arn:aws:iam::(Account A ID):role/(RoleA),arn:aws:iam::(Account B
 ID):role/(RoleB),arn:aws:iam::(Account C ID):role/(RoleC)',
    s3BucketRegion='us-east-1'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/loader \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
```

```
  -d '{
        "source" : "s3://(the target bucket name)/(the target date file name)",
        "iamRoleArn" : "arn:aws:iam::(Account A
 ID):role/(RoleA),arn:aws:iam::(Account B ID):role/(RoleB),arn:aws:iam::(Account C
 ID):role/(RoleC)",
        "format" : "csv",
        "region" : "us-east-1"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -X POST https://your-neptune-endpoint:port/loader \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://(the target bucket name)/(the target date file name)",
        "iamRoleArn" : "arn:aws:iam::(Account A
 ID):role/(RoleA),arn:aws:iam::(Account B ID):role/(RoleB),arn:aws:iam::(Account C
 ID):role/(RoleC)",
        "format" : "csv",
        "region" : "us-east-1"
      }'
```

Formats de chargement de données

L'API Amazon Neptune `Load` permet le chargement de données dans différents formats.

Property-graph formats de chargement

Les données chargées dans l'un des formats de graphes de propriétés suivants peuvent être interrogées à l'aide de Gremlin et d'openCypher :

- Format de chargement de données Gremlin (`csv`) : format CSV (valeurs séparées par une virgule).
- Format de chargement de données openCypher (`opencypher`) : format CSV (valeurs séparées par des virgules).

Formats de chargement RDF

Pour charger des données RDF (Resource Description Framework) à interroger avec SPARQL, vous pouvez utiliser l'un des formats standard suivants comme spécifié par W3C (World Wide Web Consortium) :

- N-Triples (`ntriples`) à partir de la spécification àhttps://www.w3.org/TR/n-triples/.
- N-Quads (`nquads`) à partir de la spécification àhttps://www.w3.org/TR/n-quads/.
- RDF/XML (`rdfxml`) à partir de la spécification àhttps://www.w3.org/TR/rdf-syntax-grammar/.
- Turtle (`turtle`) de la spécification à l'adresse https://www.w3.org/TR/turtle/

Les données de chargement doivent utiliser le UTF-8 codage

> **Important**
>
> Tous les fichiers de données de chargement doivent être codés sous UTF-8 forme. Si un fichier n'est pas UTF-8 codé, Neptune essaie de le charger tel UTF-8 quel.

Pour N-Quads les N-triples données contenant des caractères Unicode, les séquences d'\u*xxxxx*échappement sont prises en charge. Toutefois, Neptune ne prend pas en charge la normalisation. Si une valeur nécessite une normalisation, elle ne sera pas mise en correspondance octet à octet lors de l'interrogation. Pour plus d'informations sur la normalisation, consultez la page Normalisation sur Unicode.org.

Si vos données ne sont pas dans un format pris en charge, vous devez les convertir avant de les charger.

Un outil de conversion de GraphML au format Neptune CSV est disponible dans le projet GraphML2CSV sur. GitHub

Prise en charge de la compression des fichiers de chargement de données

Neptune prend en charge la compression des fichiers individuels au format `gzip` ou `bzip2`.

Le fichier compressé doit avoir une `.bz2` extension `.gz` ou et doit être un seul fichier texte codé au UTF-8 format. Vous pouvez charger plusieurs fichiers, mais chacun d'entre eux doit être un fichier `.gz`, `.bz2` ou non compressé distinct. Les fichiers d'archive portant des extensions telles que `.tar`, `.tar.gz` et `.tgz` ne sont pas pris en charge.

Les sections suivantes décrivent les formats de façon plus détaillée.

Rubriques

- Format de chargement de données Gremlin
- Format de chargement des données openCypher
- Formats de chargement de données RDF

Format de chargement de données Gremlin

Pour charger des données TinkerPop Apache G705 au format CSV, vous devez spécifier les sommets et les arêtes dans des fichiers séparés.

Le chargeur peut effectuer le chargement depuis plusieurs fichiers de sommet et plusieurs fichiers d'arc en une seule tâche de chargement.

Pour chaque commande de chargement, l'ensemble de fichiers à charger doit être dans le même dossier au sein du compartiment Amazon S3, et vous devez spécifier le nom du dossier pour le paramètre `source`. Les noms de fichier et les extensions de nom de fichier ne sont pas importants.

Le format CSV Amazon Neptune respecte la spécification CSV RFC 4180. Pour plus d'informations, consultez Common Format and MIME Type for CSV Files sur le site web Internet Engineering Task Force (IETF).

> **Note**
>
> Tous les fichiers doivent être encodés au UTF-8 format.

Chaque fichier comporte une ligne d'en-têtes séparés par des virgules. La ligne d'en-tête se compose d'en-têtes de colonne système et d'en-têtes de colonne de propriété.

En-têtes de colonne système

Les en-têtes de colonne système obligatoires et autorisés sont différents pour les fichiers de sommet et les fichiers d'arc.

Chaque colonne système ne peut apparaître qu'une seule fois dans un en-tête.

Toutes les étiquettes sont sensibles à la casse.

En-têtes de sommet

- `~id` - Obligatoire

  Un ID pour le sommet.
- `~label`

  Une étiquette pour le sommet. Plusieurs valeurs d'étiquette sont autorisées, séparées par des points-virgules (`;`).

  S'il n'`~label`est pas présent, TinkerPop fournit une étiquette avec la valeur`vertex`, car chaque sommet doit avoir au moins une étiquette.

En-têtes d'arc

- `~id` - Obligatoire

  Un ID pour l'arc.
- `~from` - Obligatoire

  ID de sommet du sommet from.
- `~to` - Obligatoire

  ID de sommet du sommet to.
- `~label`

  Étiquette de l'arête. Les arêtes ne peuvent avoir qu'une seule étiquette.

  S'il n'`~label`est pas présent, TinkerPop fournit une étiquette avec la valeur`edge`, car chaque arête doit avoir une étiquette.

En-têtes de colonne de propriété

Vous pouvez spécifier une colonne (`:`) pour une propriété à l'aide de la syntaxe suivante. Les noms de type ne sont pas sensibles à la casse. Notez toutefois que si le signe deux-points figure dans le nom d'une propriété, faites-le précéder d'une barre oblique inverse comme caractère d'échappement : `\:`.

```
propertyname:type
```

> **Note**
>
> Les espaces, les virgules, le retour en chariot et les caractères de nouvelle ligne ne sont pas autorisés dans les en-têtes de colonne. Les noms de propriétés ne peuvent donc pas inclure ces caractères.

Vous pouvez spécifier une colonne pour un type de tableau en ajoutant `[]` au type :

```
propertyname:type[]
```

> **Note**
>
> Les propriétés d'arc ne peuvent avoir qu'une seule valeur et provoquent une erreur si un type de tableau ou une seconde valeur est spécifié.

L'exemple suivant montre l'en-tête de colonne pour une propriété nommée `age` de type `Int`.

```
age:Int
```

Chaque ligne du fichier doit obligatoirement avoir un nombre entier dans cette position ou rester vide.

Les tableaux de chaînes sont autorisés, mais les chaînes d'un tableau ne peuvent pas inclure le point-virgule (`;`), à moins qu'une barre oblique inverse n'y soit ajoutée comme caractère d'échappement (`\;`, par exemple).

Spécification de la cardinalité d'une colonne

L'en-tête de colonne peut être utilisé pour spécifier la cardinalité pour la propriété identifiée par la colonne. Ceci permet au chargeur en bloc de respecter la cardinalité de manière similaire à la façon dont le font les requêtes Gremlin.

Vous spécifiez la cardinalité d'une colonne comme suit :

```
propertyname:type(cardinality)
```

La *`cardinality`* valeur peut être `single` soit`set`. La valeur par défaut est supposée être `set`, ce qui signifie que la colonne peut accepter plusieurs valeurs. Dans le cas des fichiers d'arête, la

cardinalité est toujours unique (« single ») et si vous spécifiez une autre cardinalité, le chargeur déclenche une exception.

Si la cardinalité est `single`, le chargeur déclenche une erreur si une valeur précédente est déjà présente pendant le chargement d'une valeur ou si plusieurs valeurs sont chargées. Ce comportement peut être ignoré afin qu'une valeur existante soit remplacée lorsqu'une nouvelle valeur est chargée à l'aide de l'indicateur `updateSingleCardinalityProperties`. Consultez Commande Loader.

Il est possible d'utiliser un paramètre de cardinalité de type tableau, même si ce n'est généralement pas nécessaire. Voici les combinaisons possibles :

- `name:type` : la cardinalité est `set`, et le contenu est à valeur unique.
- `name:type[]` : la cardinalité est `set`, et le contenu est à valeurs multiples.
- `name:type(single)` : la cardinalité est `single`, et le contenu est à valeur unique.
- `name:type(set)` : la cardinalité est `set`, ce qui est identique à la valeur par défaut, et le contenu est à valeur unique.
- `name:type(set)[]` : la cardinalité est `set`, et le contenu est à valeurs multiples.
- `name:type(single)[]` : ce paramètre est contradictoire et génère une erreur.

La section suivante répertorie tous les types de données Gremlin disponibles.

Types de données Gremlin

Il s'agit d'une liste des types de propriété autorisés, avec une description de chaque type.

Bool (ou booléen)

Indique un champ booléen. Valeurs autorisées : `false`, `true`

> **Note**
>
> Toute valeur autre que `true` sera traitée comme false.

Types de nombres entiers

Les valeurs en dehors des plages définies entraînent une erreur.

Type	Range
Octet	-128 à 127
Court	-32768 à 32767
Int	-2^31 à 2^31-1
Long	-2^63 à 2^63-1

Types de nombre décimal

Prend en charge la notation décimale ou la notation scientifique. Autorise également les symboles tels que (+/-) Infinity ou NaN. La clause INF n'est pas prise en charge.

Type	Range
Float	Virgule flottante IEEE 754 32 bits
Double	Virgule flottante IEEE 754 64 bits

Les valeurs à virgule flottante et doubles qui sont trop longues sont chargées et arrondie à la valeur la plus proche pour la précision 24 bits (virgule flottante) et 53 bits (double). Une valeur du milieu est arrondie à 0 pour le dernier chiffre restant au niveau du bit.

String

Les guillemets sont facultatifs. Les virgules, et les caractères de saut de ligne et de retour à la ligne font automatiquement l'objet d'un échappement s'ils sont inclus dans une chaîne entourée de guillemets ("). Exemple : `"Hello, World"`

Pour inclure des guillemets dans une chaîne entre guillemets, vous pouvez échapper les guillemets en utilisant deux guillemets dans une ligne : Exemple : `"Hello ""World"""`

Les tableaux de chaînes sont autorisés, mais les chaînes d'un tableau ne peuvent pas inclure le point-virgule (`;`), à moins qu'une barre oblique inverse n'y soit ajoutée comme caractère d'échappement (`\;`, par exemple).

Si vous souhaitez placer des chaînes d'un tableau entre guillemets, vous devez entourer la totalité du tableau par un ensemble de guillemets. Exemple : `"String one; String 2; String 3"`

Date

Date Java au ISO-8601 format. Supporte les formats suivants :`yyyy-MM-dd`,`yyyy-MM-ddTHH:mm`,`yyyy-MM-ddTHH:mm:ss`,`yyyy-MM-ddTHH:mm:ssZ`. Les valeurs sont converties en heure d'époque et stockées.

Datetime

Date Java au ISO-8601 format. Supporte les formats suivants :`yyyy-MM-dd`,`yyyy-MM-ddTHH:mm`,`yyyy-MM-ddTHH:mm:ss`,`yyyy-MM-ddTHH:mm:ssZ`. Les valeurs sont converties en heure d'époque et stockées.

Format de ligne Gremlin

Délimiteurs

Les champs dans une ligne sont séparés par une virgule. Les enregistrements sont séparés par un saut de ligne ou par un saut de ligne suivi d'un retour chariot.

Champs vides

Des champs vides sont autorisés pour les colonnes non obligatoires (comme des propriétés définies par l'utilisateur). Un champ vide a quand même besoin d'une virgule comme séparateur. Les champs vides sur les colonnes obligatoires entraîneront une erreur d'analyse. Les valeurs de chaîne vides sont interprétées comme des valeurs de chaîne vides pour le champ, et non comme un champ vide. L'exemple de la section suivante comporte un champ vide dans chaque exemple de sommet.

ID de sommet

Les valeurs `~id` doivent être uniques pour tous les sommets dans chaque fichier de sommets. Plusieurs lignes de sommet avec des valeurs `~id` identiques sont appliquées à un seul sommet dans le graphique. Une chaîne vide (`""`) est un identifiant valide, et le sommet est créé avec une chaîne vide comme identifiant.

ID d'arête

En outre, les valeurs `~id` doivent être uniques pour toutes les arêtes dans chaque fichier d'arête. Plusieurs lignes d'arête avec des valeurs `~id` identiques sont appliquées à la seule arête du graphe.

La chaîne vide ("") est un identifiant valide, et le bord est créé avec une chaîne vide comme identifiant.

Étiquettes

Les étiquettes distinguent les majuscules et minuscules et ne peuvent pas être vides. Une valeur de "" provoquera une erreur.

Valeurs de chaîne

Les guillemets sont facultatifs. Les virgules, et les caractères de saut de ligne et de retour à la ligne font automatiquement l'objet d'un échappement s'ils sont inclus dans une chaîne entourée de guillemets ("). ("")Les valeurs de chaîne vides sont interprétées comme une valeur de chaîne vide pour le champ, et non comme un champ vide.

Spécification du format CSV

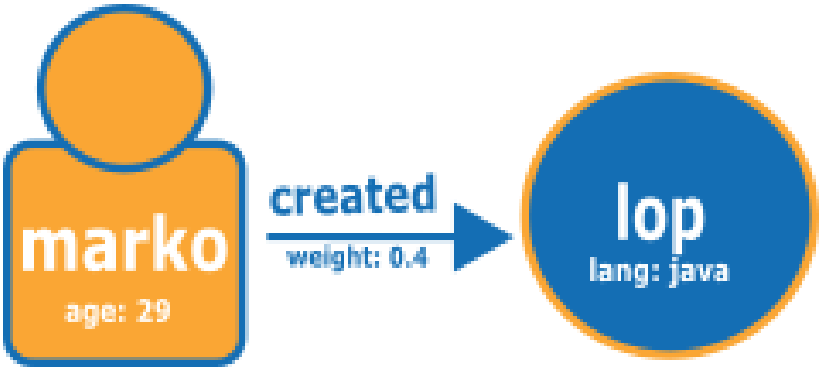
Le format CSV Neptune respecte la spécification CSV RFC 4180, y compris les exigences suivantes.

- Les fins de ligne de style Unix et Windows sont prises en charge (\n ou \r\n).
- Tout champ peut être placé entre guillemets.
- Les champs contenant un saut de ligne, des guillemets ou des virgules doivent être placés entre guillemets. (Si ce n'est pas le cas, le chargement s'interrompt immédiatement.)
- Les guillemets (") dans un champ doit être représentés par des guillemets doubles. Par exemple, une chaîne `Hello "World"` doit figurer sous la forme `"Hello ""World"""` dans les données.
- Les espaces entre les délimiteurs sont ignorés. Si une ligne est présente sous forme`value1, value2`, elle est stockée sous forme `"value1"` et`"value2"`.
- Tous les autres caractères d'échappement sont stockés tels quels. Par exemple, `"data1\tdata2"` est stocké comme suit : `"data1\tdata2"`. Aucun autre échappement n'est nécessaire dans la mesure où ces caractères sont placés entre guillemets.
- Les champs vides sont autorisés. Un champ vide est considéré comme une valeur vide.
- Plusieurs valeurs pour un champ sont spécifiées séparées par un point-virgule (`;`).

Pour plus d'informations, consultez Common Format and MIME Type for CSV Files sur le site web Internet Engineering Task Force (IETF).

Exemple Gremlin

Le schéma suivant montre un exemple de deux sommets et d'une arête extraits du graphe TinkerPop moderne.



Voici le graphe au format de chargement CSV Neptune.

Fichier de sommets :

```
~id,name:String,age:Int,lang:String,interests:String[],~label
v1,"marko",29,,"sailing;graphs",person
v2,"lop",,"java",,software
```

Vue tabulaire du fichier de sommets :

~id	name:String	age:Int	lang:String	Intérêts:string[]	~étiquette
v1	"marko"	29		["navigation », « graphes"]	personne
v2	"lop"		"java"		logiciel

Fichier d'arête :

```
~id,~from,~to,~label,weight:Double
e1,v1,v2,created,0.4
```

Vue tabulaire du fichier d'arête :

~id	~de	~sur	~étiquette	weight:Double
e1	v1	v2	créé	0.4

Étapes suivantes

Maintenant que vous en savez plus sur les formats de chargement, consultez Exemple : chargement de données dans une instance de base de données Neptune.

Format de chargement des données openCypher

Pour charger des données openCypher à l'aide du format CSV openCypher, vous devez spécifier les nœuds et les relations dans des fichiers séparés. Le chargeur peut charger les données à partir de plusieurs de ces fichiers de nœuds et fichiers de relations en une seule tâche de chargement.

Pour chaque commande de chargement, l'ensemble de fichiers à charger doit avoir le même préfixe de chemin dans un compartiment Amazon Simple Storage Service. Vous spécifiez ce préfixe dans le paramètre source. Les noms et les extensions de fichier ne sont pas importants.

Dans Amazon Neptune, le format CSV openCypher est conforme à la spécification CSV RFC 4180. Pour plus d'informations, consultez Common Format et type MIME pour les fichiers CSV (https://tools.ietf.org/html/rfc4180) sur le site Web de l'Internet Engineering Task Force (IETF).

> **Note**
>
> Ces fichiers DOIVENT être encodés au UTF-8 format.

Chaque fichier possède une ligne d'en-têtes séparés par des virgules, qui contient à la fois les en-têtes de colonne système et les en-têtes de colonne des propriétés.

En-têtes de colonne système dans les fichiers de chargement de données openCypher

Une colonne système spécifique ne peut apparaître qu'une seule fois dans chaque fichier. Toutes les étiquettes d'en-tête de colonne système sont sensibles à la casse.

Les en-têtes de colonne système obligatoires et autorisés sont différents pour les fichiers de chargement de nœuds et les fichiers de chargement de relations openCypher :

En-têtes de colonnes système dans les fichiers de nœuds

- **`:ID`** : (obligatoire) ID du nœud.

  Un espace d'ID facultatif peut être ajouté à l'en-tête de colonne du nœud `:ID` comme ceci : `:ID(`*`ID Space`*`)`. Par exemple : `:ID(movies)`.

  Lorsque vous chargez des relations qui connectent les nœuds de ce fichier, utilisez les mêmes espaces d'identification dans les `:START_ID` and/or `:END_ID` colonnes des fichiers de relations.

  La colonne du nœud `:ID` peut éventuellement être stockée en tant que propriété sous la forme *`property name`*`:ID`. Par exemple : `name:ID`.

  Les ID de nœud doivent être uniques pour tous les fichiers de nœuds des chargements actuels et précédents. Si un espace d'ID est utilisé, les ID de nœud doivent être uniques dans tous les fichiers de nœuds qui utilisent le même espace d'ID lors des chargements actuels et précédents.
- **`:LABEL`** : étiquette du nœud.

  Lorsque vous utilisez plusieurs valeurs d'étiquette pour un seul nœud, chaque étiquette doit être séparée par des points-virgules (). `;`

En-têtes de colonnes système dans les fichiers de relations

- **`:ID`** : ID de la relation. Obligatoire lorsque `userProvidedEdgeIds` est défini sur true (valeur par défaut), mais non valide quand `userProvidedEdgeIds` indique `false`.

  Les ID de relation doivent être uniques pour tous les fichiers de relations des chargements actuels et précédents.
- **`:START_ID`** : (obligatoire) ID du nœud à partir duquel cette relation commence.

  Un espace d'ID peut être associé à la colonne d'ID de départ sous la forme `:START_ID(`*`ID Space`*`)`. L'espace d'ID attribué à l'ID du nœud de départ doit correspondre à l'espace d'ID attribué au nœud dans son fichier de nœud.
- **`:END_ID`** : (obligatoire) ID du nœud où cette relation se termine.

  Un espace d'ID peut être associé à la colonne d'ID de fin sous la forme `:END_ID(`*`ID Space`*`)`. L'espace d'ID attribué à l'ID du nœud de fin doit correspondre à l'espace d'ID attribué au nœud dans son fichier de nœud.
- **`:TYPE`** : type de relation. Les relations ne peuvent avoir qu'un seul type.

> **Note**
>
> Consultez Chargement de données openCypher pour plus d'informations sur la façon dont les ID de nœud ou de relation dupliqués sont gérés par le processus de chargement en bloc.

En-têtes de colonnes de propriétés dans les fichiers de chargement de données openCypher

Vous pouvez spécifier qu'une colonne contient les valeurs d'une propriété particulière à l'aide d'un en-tête de colonne de propriété sous la forme suivante :

```
propertyname:type
```

Les espaces, les virgules, le retour en chariot et les caractères de nouvelle ligne ne sont pas autorisés dans les en-têtes de colonne. Les noms de propriétés ne peuvent donc pas inclure ces caractères. Voici un exemple d'en-tête de colonne pour une propriété nommée `age` de type `Int` :

```
age:Int
```

La colonne avec `age:Int` comme en-tête de colonne devrait alors contenir un entier ou une valeur vide sur chaque ligne.

Types de données dans les fichiers de chargement de données Neptune openCypher

- **`Bool`** ou **`Boolean`** : champ booléen. Les valeurs autorisées sont `true` et `false`.

  Toute valeur autre que `true` qui est traitée comme `false`.
- **`Byte`** : nombre entier compris entre `-128` et `127`.
- **`Short`** : nombre entier compris entre `-32,768` et `32,767`.
- **`Int`** : nombre entier compris entre `-2^31` et `2^31 - 1`.
- **`Long`** : nombre entier compris entre `-2^63` et `2^63 - 1`.
- **`Float`** : nombre à virgule flottante IEEE 754 32 bits. La notation décimale et la notation scientifique sont toutes deux prises en charge. `Infinity`, `-Infinity`, et `NaN` sont tous reconnus, mais pas `INF`.

  Les valeurs contenant trop de chiffres sont arrondies à la valeur la plus proche (une valeur intermédiaire est arrondie à 0 pour le dernier chiffre restant au niveau du bit).

- **`Double`** : nombre à virgule flottante IEEE 754 64 bits. La notation décimale et la notation scientifique sont toutes deux prises en charge. `Infinity`, `-Infinity`, et `NaN` sont tous reconnus, mais pas `INF`.

  Les valeurs contenant trop de chiffres sont arrondies à la valeur la plus proche (une valeur intermédiaire est arrondie à 0 pour le dernier chiffre restant au niveau du bit).

- **`String`** : les guillemets sont facultatifs. Les virgules et les caractères de saut de ligne et de retour à la ligne font automatiquement l'objet d'un échappement s'ils sont inclus dans une chaîne entourée de guillemets doubles (") comme `"Hello, World"`.

  Vous pouvez inclure des guillemets dans une chaîne qui contient déjà des guillemets en utilisant deux guillemets d'affilée comme `"Hello ""World"""`.

- **`DateTime`**— Une date Java dans l'un des ISO-8601 formats suivants :
  - `yyyy-MM-dd`
  - `yyyy-MM-ddTHH:mm`
  - `yyyy-MM-ddTHH:mm:ss`
  - `yyyy-MM-ddTHH:mm:ssZ`

Auto-cast types de données dans les fichiers de chargement de données Neptune OpenCypher

Auto-cast des types de données sont fournis pour charger des types de données qui ne sont pas actuellement pris en charge nativement par Neptune. Les données de ces colonnes sont stockées sous forme de chaînes, mot pour mot, sans vérification par rapport au format prévu. Voici les types de données diffusées automatiquement qui sont autorisés :

- **`Char`** : champ `Char`. Stocké sous forme de chaîne.
- **`Date`**, **`LocalDate`**, et **`LocalDateTime`** : consultez Instants temporels Neo4j pour obtenir une description des types `date`, `localdate` et `localdatetime`. Les valeurs sont chargées telles quelles sous forme de chaînes, sans validation.
- **`Duration`** : consultez le format de durée Neo4j. Les valeurs sont chargées telles quelles sous forme de chaînes, sans validation.
- Point : champ de point destiné au stockage de données spatiales. Consultez Instants spatiaux. Les valeurs sont chargées telles quelles sous forme de chaînes, sans validation.

Exemple de format de chargement openCypher

Le schéma suivant, tiré du TinkerPop Modern Graph, montre un exemple de deux nœuds et d'une relation :

Voici le graphe au format de chargement Neptune openCypher normal.

Fichier de nœud :

```
:ID,name:String,age:Int,lang:String,:LABEL
v1,"marko",29,,person
v2,"lop",,"java",software
```

Fichier de relation :

```
:ID,:START_ID,:END_ID,:TYPE,weight:Double
e1,v1,v2,created,0.4
```

Vous pouvez également utiliser les espaces d'ID et l'ID en tant que propriétés, comme suit :

Fichier du premier nœud :

```
name:ID(person),age:Int,lang:String,:LABEL
"marko",29,,person
```

Fichier du deuxième nœud :

```
name:ID(software),age:Int,lang:String,:LABEL
"lop",,"java",software
```

Fichier de relation :

```
:ID,:START_ID(person),:END_ID(software),:TYPE,weight:Double
e1,"marko","lop",created,0.4
```

Formats de chargement de données RDF

Pour charger des données RDF (Resource Description Framework), vous pouvez utiliser l'un des formats standard suivants comme spécifié par W3C (World Wide Web Consortium) :

- N-Triples (`ntriples`) à partir de la spécification à https://www.w3.org/TR/n-triples/
- N-Quads (`nquads`) à partir de la spécification à https://www.w3.org/TR/n-quads/
- RDF/XML (`rdfxml`) à partir de la spécification à https://www.w3.org/TR/rdf-syntax-grammar/
- Turtle (`turtle`) de la spécification à l'adresse https://www.w3.org/TR/turtle/

> **Important**
>
> Tous les fichiers doivent être encodés au UTF-8 format.
> Pour N-Quads les N-triples données contenant des caractères Unicode, les séquences d'\u*xxxxx*échappement sont prises en charge. Toutefois, Neptune ne prend pas en charge la normalisation. Si une valeur nécessite une normalisation, elle ne sera pas mise en correspondance octet à octet lors de l'interrogation. Pour plus d'informations sur la normalisation, consultez la page Normalisation sur Unicode.org.

Étapes suivantes

Maintenant que vous en savez plus sur les formats de chargement, consultez Exemple : chargement de données dans une instance de base de données Neptune.

Exemple : chargement de données dans une instance de base de données Neptune

Cet exemple montre comment charger des données dans Amazon Neptune. Sauf indication contraire, vous devez suivre ces étapes à partir d'une instance Amazon Elastic Compute Cloud (Amazon EC2) située dans le même Amazon Virtual Private Cloud (VPC) que l'instance de base de données Neptune.

Prérequis pour l'exemple de chargement de données

Avant de commencer, les prérequis suivants doivent être remplis :

- Vous devez disposer d'une instance de base de données Neptune.

  Pour plus d'informations sur le lancement d'une instance de base de données Neptune, consultez Création d'un cluster Amazon Neptune.
- Vous devez disposer d'un compartiment Amazon Simple Storage Service (Amazon S3) dans lequel stocker les fichiers de données :

  Vous pouvez utiliser un compartiment existant. Si vous n'avez pas de compartiment S3, consultez Création d'un compartiment dans le Guide de démarrage Amazon S3.
- Vous devez disposer des données de graphe à charger, dans un des formats compatibles avec le chargeur :

  Si vous utilisez Gremlin pour interroger le graphe, Neptune peut charger les données dans un format de valeurs séparées par des virgules (`CSV`), comme décrit dans Format de chargement de données Gremlin.

  Si vous utilisez OpenCypher pour interroger votre graphe, Neptune peut également charger des données dans un Cypher-specific `CSV` format ouvert, comme décrit dans. Format de chargement des données openCypher

  Si vous utilisez SPARQL, Neptune peut charger les données dans divers formats RDF, comme décrit dans Formats de chargement de données RDF.
- Vous devez disposer d'un rôle IAM pour l'instance de base de données Neptune, avec une politique IAM qui autorise l'accès aux fichiers de données dans le compartiment S3. La stratégie doit accorder des autorisations de lecture et de type Liste.

  Pour plus d'informations sur la création d'un rôle ayant accès à Amazon S3 et sur son association à un cluster Neptune, consultez Prérequis : rôle IAM et accès à Amazon S3.

  > **Note**
  >
  > L'API `Load` Neptune nécessite un accès en lecture aux fichiers de données uniquement. La politique IAM n'a pas besoin d'autoriser un accès en écriture ou un accès au compartiment complet.

- Vous devez disposer d'un point de terminaison de VPC Amazon S3. Pour plus d'informations, consultez la section Création d'un point de terminaison de VPC Amazon S3.

Création d'un point de terminaison de VPC Amazon S3

Le chargeur Neptune a besoin d'un point de terminaison d'un VPC pour Amazon S3.

Pour configurer l'accès à Amazon S3

1. Connectez-vous à la console Amazon VPC AWS Management Console et ouvrez-la à l'adresse. https://console.aws.amazon.com/vpc/
2. Dans le panneau de navigation de gauche, sélectionnez Points de terminaison.
3. Choisissez Créer un point de terminaison.
4. Pour Nom du service, choisissez `com.amazonaws.`*`region`*`.s3`.

   > **Note**
   >
   > Si la région dans ce champ est incorrecte, assurez-vous que la région de la console est correcte.

5. Choisissez le VPC qui contient votre instance de base de données Neptune.
6. Cochez la case en regard des tables de routage associées aux sous-réseaux liés à votre cluster. Si vous n'avez qu'une seule table de routage, vous devez cocher cette case.
7. Choisissez Créer un point de terminaison.

Pour plus d'informations sur la création du point de terminaison, consultez Points de terminaison de VPC dans le Guide de l'utilisateur Amazon VPC. Pour plus d'informations sur les limites qui s'appliquent aux points de terminaison d'un VPC, consultez Points de terminaisons de VPC pour Amazon S3.

Pour charger des données dans une instance de base de données Neptune

1. Copiez les fichiers de données dans un compartiment Amazon S3. Le compartiment S3 doit se trouver dans la même AWS région que le cluster qui charge les données.

   Vous pouvez utiliser la AWS CLI commande suivante pour copier les fichiers dans le compartiment.

> **Note**
>
> Cette commande n'a pas besoin d'être exécutée à partir de l'instance Amazon EC2.

```
aws s3 cp data-file-name s3://bucket-name/object-key-name
```

> **Note**
>
> Dans Amazon S3, le nom de la clé d'objet est le chemin d'accès complet d'un fichier, y compris le nom du fichier.
> Exemple : Dans la commande `aws s3 cp datafile.txt s3://examplebucket/mydirectory/datafile.txt`, le nom de la clé d'objet est **`mydirectory/datafile.txt`**.

Vous pouvez également utiliser le AWS Management Console pour télécharger des fichiers dans le compartiment S3. Ouvrez la console Amazon S3 à https://console.aws.amazon.com/s3/l'adresse et choisissez un compartiment. Dans le coin supérieur gauche, choisissez Upload (Charger) pour charger les fichiers.

2. Dans une fenêtre de ligne de commande, entrez ce qui suit pour exécuter le chargeur Neptune en utilisant les valeurs correctes pour votre point de terminaison, le chemin Amazon S3, le format et l'ARN du rôle IAM.

   Le paramètre `format` peut avoir l'une des valeurs suivantes : `csv` pour Gremlin, `opencypher` pour openCypher ou `ntriples`, `nquads`, `turtle`, et `rdfxml` pour RDF. Pour plus d'informations sur les autres paramètres, consultez Commande de chargeur Neptune.

   Consultez la section Connexion aux points de terminaison Amazon Neptune pour découvrir comment trouver le nom d'hôte de votre instance de base de données Neptune.

   Le paramètre de région doit correspondre à la région du cluster et au compartiment S3.

   Amazon Neptune est disponible dans les régions suivantes : AWS

   - USA Est (Virginie du Nord) : `us-east-1`
   - USA Est (Ohio) : `us-east-2`

- USA Ouest (Californie du Nord) : `us-west-1`
- USA Ouest (Oregon) : `us-west-2`
- Canada (Centre) : `ca-central-1`
- Canada Ouest (Calgary) : `ca-west-1`
- Amérique du Sud (São Paulo) : `sa-east-1`
- Europe (Stockholm) : `eu-north-1`
- Europe (Espagne) : `eu-south-2`
- Europe (Irlande) : `eu-west-1`
- Europe (Londres) : `eu-west-2`
- Europe (Paris) : `eu-west-3`
- Europe (Francfort) : `eu-central-1`
- Moyen-Orient (Bahreïn) : `me-south-1`
- Moyen-Orient (EAU) : `me-central-1`
- Israël (Tel Aviv) : `il-central-1`
- Afrique (Le Cap) : `af-south-1`
- Asie-Pacifique (Hong Kong) : `ap-east-1`
- Asie-Pacifique (Tokyo) : `ap-northeast-1`
- Asie-Pacifique (Séoul) : `ap-northeast-2`
- Asie-Pacifique (Osaka) : `ap-northeast-3`
- Asie-Pacifique (Singapour) : `ap-southeast-1`
- Asie-Pacifique (Sydney) : `ap-southeast-2`
- Asie-Pacifique (Jakarta) : `ap-southeast-3`
- Asie-Pacifique (Melbourne) : `ap-southeast-4`
- Asie-Pacifique (Malaisie) : `ap-southeast-5`
- Asie-Pacifique (Mumbai) : `ap-south-1`
- Asie-Pacifique (Hyderabad) : `ap-south-2`
- Chine (Beijing) : `cn-north-1`
- Chine (Ningxia) : `cn-northwest-1`
- AWS GovCloud (US-West): `us-gov-west-1`
- AWS GovCloud (US-East): `us-gov-east-1`

AWS CLI

```
aws neptunedata start-loader-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --source "s3://bucket-name/object-key-name" \
  --format "format" \
  --iam-role-arn "arn:aws:iam::account-id:role/role-name" \
  --s3-bucket-region "region" \
  --no-fail-on-error \
  --parallelism "MEDIUM" \
  --no-update-single-cardinality-properties \
  --queue-request \
  --dependencies "load_A_id" "load_B_id"
```

Pour plus d'informations, consultez start-loader-job dans la AWS CLI référence des commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_loader_job(
    source='s3://bucket-name/object-key-name',
    format='format',
    iamRoleArn='arn:aws:iam::account-id:role/role-name',
    s3BucketRegion='region',
    failOnError=False,
    parallelism='MEDIUM',
    updateSingleCardinalityProperties=False,
    queueRequest=True,
    dependencies=['load_A_id', 'load_B_id']
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/loader \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://bucket-name/object-key-name",
        "format" : "format",
        "iamRoleArn" : "arn:aws:iam::account-id:role/role-name",
        "region" : "region",
        "failOnError" : "FALSE",
        "parallelism" : "MEDIUM",
        "updateSingleCardinalityProperties" : "FALSE",
        "queueRequest" : "TRUE",
        "dependencies" : ["load_A_id", "load_B_id"]
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -X POST https://your-neptune-endpoint:port/loader \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://bucket-name/object-key-name",
        "format" : "format",
        "iamRoleArn" : "arn:aws:iam::account-id:role/role-name",
        "region" : "region",
        "failOnError" : "FALSE",
        "parallelism" : "MEDIUM",
        "updateSingleCardinalityProperties" : "FALSE",
        "queueRequest" : "TRUE",
        "dependencies" : ["load_A_id", "load_B_id"]
```

```
    }'
```

Pour plus d'informations sur la création et l'association d'un rôle IAM à un cluster Neptune, consultez Prérequis : rôle IAM et accès à Amazon S3.

> **Note**
>
> Pour de plus amples informations sur les paramètres de demande de chargement, veuillez consulter Paramètres de demande du chargeur Neptune. En bref :
> Le paramètre `source` accepte un URI Amazon S3 qui pointe vers un seul fichier ou vers un dossier. Si vous spécifiez un dossier, Neptune y charge chaque fichier de données. Le dossier peut contenir plusieurs fichiers de sommet et plusieurs fichiers d'arc.
> L'URI peut être à l'un des formats suivants.
>
> - `s3://bucket_name/object-key-name`
> - `https://s3.amazonaws.com/bucket_name/object-key-name`
> - `https://s3-us-east-1.amazonaws.com/bucket_name/object-key-name`
>
> Ce paramètre `format` peut être l'un des suivants :
>
> - Format CSV Gremlin (`csv`) pour les graphes de propriétés Gremlin
> - Format CSV openCypher (`opencypher`) pour les graphes de propriétés openCypher
> - Format N -Triples (`ntriples`) pour RDF/SPARQL
> - N-Quads (`nquads`) format pour RDF/SPARQL
> - RDF/XML (`rdfxml`) format pour RDF/SPARQL
> - Format Turtle (`turtle`) pour RDF/SPARQL
>
> Le paramètre facultatif `parallelism` vous permet de limiter le nombre de threads utilisés dans le processus de chargement en bloc. Il peut être défini sur `LOW`, `MEDIUM`, `HIGH` ou `OVERSUBSCRIBE`.
> Lorsque la valeur `updateSingleCardinalityProperties` est définie sur `"FALSE"`, le chargeur renvoie une erreur si plusieurs valeurs sont fournies dans un fichier source en cours de chargement pour une propriété de sommet d'une arête ou d'une seule cardinalité.

Si `queueRequest` est défini sur `"TRUE"`, la demande de chargement est placée dans une file d'attente lorsqu'une tâche de chargement est déjà en cours d'exécution.
Avec le paramètre `dependencies`, l'exécution de la demande de chargement dépend de la réussite d'une ou de plusieurs tâches de chargement qui ont déjà été placées dans la file d'attente.

3. Le chargeur Neptune renvoie un `id` de tâche qui vous permet de vérifier le statut ou d'annuler le processus de chargement, par exemple :

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

4. Saisissez la commande suivante pour obtenir le statut du chargement avec le paramètre `loadId` de l'étape 3 :

   AWS CLI

   ```
   aws neptunedata get-loader-job-status \
     --endpoint-url https://your-neptune-endpoint:port \
     --load-id ef478d76-d9da-4d94-8ff1-08d9d4863aa5
   ```

   Pour plus d'informations, consultez get-loader-job-status dans la référence des commandes. AWS CLI

   SDK

   ```
   import boto3
   from botocore.config import Config

   client = boto3.client(
       'neptunedata',
       endpoint_url='https://your-neptune-endpoint:port',
       config=Config(read_timeout=None, retries={'total_max_attempts': 1})
   )

   response = client.get_loader_job_status(
       loadId='ef478d76-d9da-4d94-8ff1-08d9d4863aa5'
   ```

```
)

print(response)
```

awscurl

```
awscurl 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5' \
  --region us-east-1 \
  --service neptune-db
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -G 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5'
```

Si le statut du chargement signale une erreur, vous pouvez demander un statut plus détaillé et une liste des erreurs. Pour plus d'informations et d'exemples, consultez API Neptune Loader Get-Status.

5. (Facultatif) Annulez la tâche `Load`.

   Saisissez la commande suivante pour supprimer (`Delete`) la tâche de chargement (loader) avec l'`id` de tâche de l'étape 3 :

   AWS CLI

```
aws neptunedata cancel-loader-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --load-id ef478d76-d9da-4d94-8ff1-08d9d4863aa5
```

Pour plus d'informations, consultez cancel-loader-job dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.cancel_loader_job(
    loadId='ef478d76-d9da-4d94-8ff1-08d9d4863aa5'
)

print(response)
```

awscurl

```
awscurl 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5' \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -X DELETE 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5'
```

La commande `DELETE` renvoie le code HTTP `200 OK` lorsque l'annulation a réussi.

Les données des fichiers de la tâche de chargement qui ont fini de se charger ne sont pas annulées. Les données restent dans l'instance de base de données Neptune.

Optimisation d'un chargement en bloc sur Amazon Neptune

Utilisez les stratégies suivantes pour réduire au minimum le temps d'un chargement en bloc Neptune :

- Nettoyez vos données :
  - Assurez-vous de convertir vos données dans un format de données compatible avant de les charger.
  - Supprimez les doublons ou les erreurs connues.
  - Réduisez le nombre de prédicats uniques (tels que les propriétés des arêtes et des sommets) autant que possible.
- Optimisez vos fichiers :
  - Si vous chargez des fichiers volumineux tels que des fichiers CSV à partir d'un compartiment Amazon S3, le chargeur gère pour vous la simultanéité en les analysant en fragments qu'il peut charger en parallèle. L'utilisation d'un très grand nombre de petits fichiers peut ralentir ce processus.
  - Si vous chargez plusieurs fichiers à partir d'un préfixe Amazon S3, le chargeur charge automatiquement d'abord les fichiers de sommets, puis les fichiers de périphérie. Toutefois, si vous savez que vous ne chargerez que des fichiers de périphérie, vous `edgeOnlyLoad` pouvez configurer cette option `TRUE` pour ignorer la première passe au cours de laquelle tous les fichiers sont analysés afin de déterminer leur contenu (sommets ou arêtes) afin que tous les fichiers de sommets trouvés soient chargés avant les fichiers de périphérie. Cela peut accélérer considérablement le temps de chargement, en particulier lorsque de nombreux fichiers Edge sont impliqués. Si certains fichiers de sommets sont également présents dans le même préfixe (`source`paramètre) Amazon S3, ils seront chargés mais sans aucune garantie de commande par rapport aux autres fichiers. En outre, si certains `from` ou plusieurs `to` sommets ne sont pas présents dans la base de données, l'insertion du bord peut signaler des erreurs dans le message`FROM_OR_TO_VERTEX_ARE_MISSING`. Il est recommandé de placer les nœuds et les arêtes dans un préfixe Amazon S3 distinct.

- Vérifiez les paramètres du chargeur :
  - Si vous n'avez pas besoin d'effectuer d'autres opérations pendant le chargement, utilisez le paramètre `parallelism` `OVERSUBSCRIBE`. Ce paramètre oblige le chargeur en bloc à utiliser toutes les ressources de CPU disponibles lors de son exécution. Il faut généralement 60 % à 70 % de la capacité du processeur pour que l'opération fonctionne aussi I/O rapidement que les contraintes le permettent.

    > **Note**
    >
    > Lorsque `parallelism` est défini sur `OVERSUBSCRIBE` ou `HIGH` (paramètre par défaut), le chargement des données openCypher risque de provoquer une condition de concurrence et un blocage des threads, ce qui entraîne une erreur `LOAD_DATA_DEADLOCK`. Dans ce cas, définissez `parallelism` sur une valeur inférieure et réessayez le chargement.

  - Si la tâche de chargement inclut plusieurs demandes de chargement, utilisez le paramètre `queueRequest`. Définir `queueRequest` sur `TRUE` permet à Neptune de mettre en file d'attente les demandes afin que vous n'ayez pas à attendre que l'une soit terminée avant d'en émettre une autre.
  - Si les demandes de chargement sont mises en file d'attente, vous pouvez définir des niveaux de dépendance à l'aide du paramètre `dependencies`, de sorte que l'échec d'une tâche entraîne l'échec des tâches dépendantes. Cette approche permet d'éviter toute incohérence des données chargées.
  - Si une tâche de chargement implique la mise à jour de valeurs précédemment chargées, veillez à définir le paramètre `updateSingleCardinalityProperties` sur `TRUE`. Dans le cas contraire, le chargeur traitera une tentative de mise à jour d'une valeur de cardinalité unique existante comme une erreur. Pour les données Gremlin, la cardinalité est également spécifiée dans les en-têtes des colonnes de propriétés (voir En-têtes de colonne de propriété).

    > **Note**
    >
    > Le paramètre `updateSingleCardinalityProperties` n'est pas disponible pour les données RDF (Resource Description Framework).

  - Vous pouvez utiliser le paramètre `failOnError` pour déterminer si les opérations de chargement en bloc doivent échouer ou se poursuivre en cas d'erreur. Vous pouvez également utiliser le paramètre `mode` pour vous assurer qu'une tâche de chargement reprendra le

chargement à partir du point où une tâche précédente a échoué plutôt que de traiter à nouveau des données déjà chargées.

- Augmentation de la capacité : configurez la taille maximale de l'instance d'enregistreur de votre cluster de bases de données avant le chargement en bloc. Notez que si vous procédez de la sorte, vous devez également augmenter la capacité de toutes les instances de réplica en lecture dans le cluster de bases de données ou les supprimer jusqu'à ce que vous ayez fini de charger les données.

  Lorsque le chargement en bloc est terminé, veillez à réduire à nouveau la taille de l'instance d'enregistreur.

> **Important**
>
> Si vous êtes confronté à un cycle de redémarrages répétés des réplicas en lecture en raison d'un retard de réplication lors d'un chargement en bloc, les réplicas ne parviendront probablement pas à suivre le rythme de l'enregistreur du cluster de bases de données. Vous pouvez soit mettre à l'échelle les lecteurs pour qu'ils soient plus grands que l'enregistreur, soit les supprimer temporairement pendant le chargement en bloc, puis les recréer une fois le chargement terminé.

Consultez Paramètres de demande pour plus d'informations sur la définition des paramètres de demande du chargeur.

Référence du chargeur Neptune

Cette section décrit les API `Loader` pour Amazon Neptune qui sont disponibles à partir du point de terminaison HTTP d'une instance de base de données Neptune.

> **Note**
>
> Consultez Message d'erreur et de flux liés au chargeur Neptune pour obtenir la liste des messages d'erreur et de flux renvoyés par le chargeur en cas d'erreur.

Table des matières

- Commande de chargeur Neptune

- Syntaxe des demandes du chargeur Neptune
- Paramètres de demande du chargeur Neptune
  - Considérations spéciales relatives au chargement de données openCypher
- Syntaxe de la réponse du chargeur Neptune
- Erreurs du chargeur Neptune
- Exemples de chargeur Neptune

- API Neptune Loader Get-Status
  - Demandes de Neptune Loader Get-Status
    - Syntaxe de Get-Status demande du chargeur
    - Paramètres de demande du Neptune Loader Get-Status
  - Réponses du Neptune Loader Get-Status
    - Disposition JSON du Neptune Loader Get-Status Response
    - Objets de réponse Neptune Loader Get-Status OverallStatus et FailedFeeds
    - Objet de réponse aux Get-Status erreurs de Neptune Loader
    - Objet de réponse Get-Status ErrorLogs de Neptune Loader
  - Exemples de chargeurs Neptune Get-Status
    - Exemple de demande de statut de chargement
    - Exemple de demande d'ID de chargement
    - Exemple de demande de statut de détaillé
  - Exemples de journaux d'erreurs du Neptune Loader Get-Status
    - Exemple de réponse de statut détaillée en cas d'erreur
    - Exemple d'erreur d'interruption d'une tâche de prélecture des données

- Tâche d'annulation du chargeur Neptune
  - Syntaxe de la demande d'annulation de tâche
  - Paramètres de demande d'annulation de tâche
  - Syntaxe de la réponse d'annulation de tâche
  - Erreurs d'annulation de tâche
  - Messages d'erreur d'annulation de tâche
  - Exemples d'annulation de tâche

Commande de chargeur Neptune

Charge les données d'un compartiment Amazon S3 dans une instance de base de données Neptune.

Pour charger des données, vous devez envoyer une demande `POST` HTTP au point de terminaison `https://your-neptune-endpoint:port/loader`. Les paramètres de la `loader` demande peuvent être envoyés dans le `POST` corps ou sous forme de URL-encoded paramètres.

> **Important**
>
> Le type MIME doit être `application/json`.

Le compartiment Amazon S3 doit se trouver dans la même AWS région que le cluster.

> **Note**
>
> Vous pouvez charger des données chiffrées à partir d'Amazon S3 si elles ont été chiffrées avec le mode `SSE-S3` d'Amazon S3. Dans ce cas, Neptune peut emprunter vos informations d'identification et émettre des appels `s3:getObject` en votre nom.
> Vous pouvez également charger les données chiffrées à l'aide du mode `SSE-KMS` à partir d'Amazon S3, à condition que votre rôle IAM implique les autorisations nécessaires pour accéder à AWS KMS. Sans AWS KMS autorisations appropriées, l'opération de chargement en bloc échoue et renvoie une `LOAD_FAILED` réponse.
> Neptune ne permet actuellement pas le chargement des données chiffrées Amazon S3 avec le mode `SSE-C`.

Il n'est pas nécessaire d'attendre la fin d'une tâche de chargement avant d'en commencer une autre. Neptune peut mettre en file d'attente jusqu'à 64 demandes de tâche à la fois, si leurs paramètres `queueRequest` sont définis sur `"TRUE"`. L'ordre des tâches en file d'attente repose sur une structure FIFO (premier entré, premier sorti). En revanche, si vous ne voulez pas qu'une tâche de chargement soit mise en file d'attente, vous pouvez définir son paramètre `queueRequest` sur `"FALSE"` (valeur par défaut), de sorte que la tâche de chargement échoue si une autre est déjà en cours.

Vous pouvez utiliser le paramètre `dependencies` pour mettre en file d'attente une tâche qui doit être exécutée uniquement une fois que les tâches précédentes spécifiées dans la file d'attente se sont achevées correctement. Si vous faites cela et que l'une des

tâches spécifiées échoue, votre tâche ne sera pas exécutée et son statut sera défini sur `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

Syntaxe des demandes du chargeur Neptune

```
{
  "source" : "string",
  "format" : "string",
  "iamRoleArn" : "string",
  "mode": "NEW|RESUME|AUTO",
  "region" : "us-east-1",
  "failOnError" : "string",
  "parallelism" : "string",
  "parserConfiguration" : {
    "baseUri" : "http://base-uri-string",
    "namedGraphUri" : "http://named-graph-string"
  },
  "updateSingleCardinalityProperties" : "string",
  "queueRequest" : "TRUE",
  "dependencies" : ["load_A_id", "load_B_id"]
}
```

OnlyLoad Syntaxe edge

Pour an`edgeOnlyLoad`, la syntaxe serait la suivante :

```
{
"source" : "string",
"format" : "string",
"iamRoleArn" : "string",
"mode": "NEW|RESUME|AUTO",
"region" : "us-east-1",
"failOnError" : "string",
"parallelism" : "string",
"edgeOnlyLoad" : "string",
"parserConfiguration" : {
    "baseUri" : "http://base-uri-string",
    "namedGraphUri" : "http://named-graph-string"
},
"updateSingleCardinalityProperties" : "string",
"queueRequest" : "TRUE",
"dependencies" : ["load_A_id", "load_B_id"]
}
```

Paramètres de demande du chargeur Neptune

- **`source`** : URI Amazon S3.

  Le paramètre `SOURCE` accepte un URI Amazon S3 qui identifie un seul fichier, plusieurs fichiers, un dossier ou plusieurs dossiers. Neptune charge tous les fichiers de données dans n'importe quel dossier spécifié.

  L'URI peut être à l'un des formats suivants.

  - `s3://`*`bucket_name`*`/`*`object-key-name`*
  - `https://s3.amazonaws.com/`*`bucket_name`*`/`*`object-key-name`*
  - `https://s3.us-east-1.amazonaws.com/`*`bucket_name`*`/`*`object-key-name`*

  L'`object-key-name`élément de l'URI est équivalent au paramètre de préfixe dans un appel d'ListObjectsAPI Amazon S3. Il identifie tous les objets du compartiment Amazon S3 spécifié dont le nom commence par ce préfixe. Il peut s'agir d'un seul fichier ou dossier, ou de plusieurs and/or dossiers de fichiers.

  Le dossier spécifié peut contenir plusieurs fichiers de sommet et plusieurs fichiers d'arête.

  Par exemple, si vous aviez la structure de dossiers et les fichiers suivants dans un compartiment Amazon S3 nommé `bucket-name` :

  ```
  s3://bucket-name/a/bc
  s3://bucket-name/ab/c
  s3://bucket-name/ade
  s3://bucket-name/bcd
  ```

  Si le paramètre source est spécifié comme`s3://bucket-name/a`, les trois premiers fichiers seront chargés.

  ```
  s3://bucket-name/a/bc
  s3://bucket-name/ab/c
  s3://bucket-name/ade
  ```

- **`format`** : format des données. Pour plus d'informations sur les formats de données pour la commande `Loader` Neptune, consultez Utilisation du chargeur en vrac Amazon Neptune pour ingérer des données.

Valeurs autorisées

- **`csv`** pour le format de données CSV Gremlin.
- **`opencypher`** pour le format de données CSV openCypher.
- **`ntriples`**pour le format de données N-Triples RDF.
- **`nquads`**pour le format de données N-Quads RDF.
- **`rdfxml`** pour le format de données RDF RDF/XML.
- **`turtle`** pour le format de données RDF Turtle.

- **`iamRoleArn`** : Amazon Resource Name (ARN) d'un rôle IAM qui doit être endossé par l'instance de base de données Neptune pour accéder au compartiment S3. Pour plus d'informations sur la création d'un rôle ayant accès à Amazon S3 et sur son association à un cluster Neptune, consultez Prérequis : rôle IAM et accès à Amazon S3.

  À partir de la version 1.2.1.0.R3 du moteur, vous pouvez également enchaîner plusieurs rôles IAM si l'instance de base de données Neptune et le compartiment Amazon S3 se trouvent dans des comptes différents. AWS Dans ce cas, `iamRoleArn` contient une liste d'ARN de rôles séparés par des virgules, comme décrit dans Chaînage des rôles IAM dans Amazon Neptune. Par exemple :

  AWS CLI

  ```
  aws neptunedata start-loader-job \
    --endpoint-url https://your-neptune-endpoint:port \
    --source "s3://(the target bucket name)/(the target date file name)" \
    --format "csv" \
    --iam-role-arn "arn:aws:iam::(Account A ID):role/(RoleA),arn:aws:iam::(Account B
   ID):role/(RoleB),arn:aws:iam::(Account C ID):role/(RoleC)" \
    --s3-bucket-region "us-east-1"
  ```

  Pour plus d'informations, consultez start-loader-job dans la AWS CLI référence des commandes.

  SDK

  ```
  import boto3
  from botocore.config import Config

  client = boto3.client(
      'neptunedata',
      endpoint_url='https://your-neptune-endpoint:port',
  ```

```
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_loader_job(
    source='s3://(the target bucket name)/(the target date file name)',
    format='csv',
    iamRoleArn='arn:aws:iam::(Account A ID):role/(RoleA),arn:aws:iam::(Account B
 ID):role/(RoleB),arn:aws:iam::(Account C ID):role/(RoleC)',
    s3BucketRegion='us-east-1'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/loader \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://(the target bucket name)/(the target date file name)",
        "iamRoleArn" : "arn:aws:iam::(Account A
 ID):role/(RoleA),arn:aws:iam::(Account B ID):role/(RoleB),arn:aws:iam::(Account C
 ID):role/(RoleC)",
        "format" : "csv",
        "region" : "us-east-1"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -X POST https://your-neptune-endpoint:port/loader \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://(the target bucket name)/(the target date file name)",
```

```
       "iamRoleArn" : "arn:aws:iam::(Account A
ID):role/(RoleA),arn:aws:iam::(Account B ID):role/(RoleB),arn:aws:iam::(Account C
ID):role/(RoleC)",
       "format" : "csv",
       "region" : "us-east-1"
     }'
```

- **`region`**— Le `region` paramètre doit correspondre à la AWS région du cluster et au compartiment S3.

  Amazon Neptune est disponible dans les régions suivantes :

  - USA Est (Virginie du Nord) : `us-east-1`
  - USA Est (Ohio) : `us-east-2`
  - USA Ouest (Californie du Nord) : `us-west-1`
  - USA Ouest (Oregon) : `us-west-2`
  - Canada (Centre) : `ca-central-1`
  - Canada Ouest (Calgary) : `ca-west-1`
  - Amérique du Sud (São Paulo) : `sa-east-1`
  - Europe (Stockholm) : `eu-north-1`
  - Europe (Espagne) : `eu-south-2`
  - Europe (Irlande) : `eu-west-1`
  - Europe (Londres) : `eu-west-2`
  - Europe (Paris) : `eu-west-3`
  - Europe (Francfort) : `eu-central-1`
  - Moyen-Orient (Bahreïn) : `me-south-1`
  - Moyen-Orient (EAU) : `me-central-1`
  - Israël (Tel Aviv) : `il-central-1`
  - Afrique (Le Cap) : `af-south-1`
  - Asie-Pacifique (Hong Kong) : `ap-east-1`
  - Asie-Pacifique (Tokyo) : `ap-northeast-1`
  - Asie-Pacifique (Séoul) : `ap-northeast-2`
  - Asie-Pacifique (Osaka) : `ap-northeast-3`
  - Asie-Pacifique (Singapour) : `ap-southeast-1`

- Asie-Pacifique (Sydney) : `ap-southeast-2`
- Asie-Pacifique (Jakarta) : `ap-southeast-3`
- Asie-Pacifique (Melbourne) : `ap-southeast-4`
- Asie-Pacifique (Malaisie) : `ap-southeast-5`
- Asie-Pacifique (Mumbai) : `ap-south-1`
- Asie-Pacifique (Hyderabad) : `ap-south-2`
- Chine (Beijing) : `cn-north-1`
- Chine (Ningxia) : `cn-northwest-1`
- AWS GovCloud (US-West): `us-gov-west-1`
- AWS GovCloud (US-East): `us-gov-east-1`

- **`mode`** : mode de la tâche de chargement.

  Valeurs autorisées : `RESUME`, `NEW`, `AUTO`

  Valeur par défaut : `AUTO`

  - `RESUME` : en mode REPRISE, le chargeur recherche un chargement précédent à partir de cette source et, s'il en trouve un, reprend cette tâche de chargement. Si aucune tâche de chargement précédente n'est trouvée, le chargeur s'arrête.

    Le chargeur évite de recharger les fichiers qui ont été chargés avec succès lors d'une tâche précédente. Il essaie uniquement de traiter les fichiers ayant échoué. Si vous aviez supprimé des données précédemment chargées à partir de votre cluster Neptune, ces données ne sont pas rechargées dans ce mode. Si une tâche de chargement précédente a chargé tous les fichiers de la même source avec succès, rien n'est rechargé, et le chargeur renvoie une réussite.
  - `NEW` : le mode NOUVEAU crée une autre demande de chargement sans tenir compte des chargements précédents. Vous pouvez utiliser ce mode pour recharger toutes les données d'une source après la suppression de données précédemment chargées à partir de votre cluster Neptune ou pour charger de nouvelles données disponibles sur la même source.
  - `AUTO` : en mode AUTO, le chargeur recherche une tâche de chargement précédente à partir de la même source, et s'il en trouve une, reprend cette tâche, exactement comme en mode `RESUME`.

    Si le chargeur ne trouve pas de tâche de chargement précédente à partir de la même source, il charge toutes les données de la source, exactement comme en mode `NEW`.

- **`edgeOnlyLoad`**— Un drapeau qui contrôle l'ordre de traitement des fichiers lors du chargement en masse.

  Valeurs autorisées : `"TRUE"`, `"FALSE"`.

  Valeur par défaut : `"FALSE"`.

  Lorsque ce paramètre est défini sur « FALSE », le chargeur charge automatiquement les fichiers de sommets en premier, puis les fichiers de bords. Pour ce faire, il analyse d'abord tous les fichiers pour déterminer leur contenu (sommets ou arêtes). Lorsque ce paramètre est défini sur « TRUE », le chargeur ignore la phase de numérisation initiale et charge immédiatement tous les fichiers dans l'ordre dans lequel ils apparaissent. Pour plus d'informations, voir Optimisation du chargement en vrac.

- **`failOnError`** : indicateur permettant d'activer un arrêt complet au niveau d'une erreur.

  Valeurs autorisées : `"TRUE"`, `"FALSE"`.

  Valeur par défaut : `"TRUE"`.

  Lorsque ce paramètre est défini sur `"FALSE"`, le chargeur essaie de charger toutes les données à l'emplacement spécifié, en ignorant les entrées contenant des erreurs.

  Lorsque ce paramètre est défini sur `"TRUE"`, le chargeur s'arrête dès qu'il rencontre une erreur. Les données chargées jusqu'à ce point persistent.

- **`parallelism`** : paramètre facultatif qui peut être défini de façon à réduire le nombre de threads utilisés par le processus de chargement en bloc.

  Valeurs autorisées :

  - `LOW` : le nombre de threads utilisés est le nombre de vCPU disponibles divisé par huit.
  - `MEDIUM` : le nombre de threads utilisés est le nombre de vCPU disponibles divisé par deux.
  - `HIGH` : le nombre de threads utilisés est le même que le nombre de vCPU disponibles.
  - `OVERSUBSCRIBE` : le nombre de threads utilisés est le nombre de vCPU disponibles multiplié par deux. Si cette valeur est utilisée, le chargeur en bloc accepte toutes les ressources disponibles.

    Cela ne signifie toutefois pas que le paramètre `OVERSUBSCRIBE` entraîne une utilisation du CPU à 100 %. L'opération de chargement étant limitée, I/O l'utilisation maximale du processeur à laquelle on peut s'attendre se situe entre 60 % et 70 %.

Valeur par défaut : `HIGH`

Le paramètre `parallelism` peut parfois entraîner un blocage entre les threads lors du chargement des données openCypher. Lorsque cela se produit, Neptune renvoie le message d'erreur `LOAD_DATA_DEADLOCK`. Vous pouvez généralement résoudre le problème en définissant un paramètre `parallelism` inférieur et en soumettant la commande de chargement à une nouvelle tentative.

- **`parserConfiguration`** : objet facultatif avec des valeurs de configuration d'analyseur supplémentaires. Chacun des paramètres enfants est également facultatif :

Nom	Exemple de valeur	Description
`namedGraphUri`	*`http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`*	Graphe par défaut pour tous les formats RDF quand aucun graphe n'est spécifié (pour les formats non-quads et les entrées NQUAD sans graphe). La valeur par défaut est `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph` .
`baseUri`	*`http://aws.amazon.com/neptune/default`*	L'URI de base pour les formats RDF/XML et Turtle. La valeur par défaut est `http://aws.amazon.com/neptune/default` .
`allowEmptyStrings`	*`true`*	Les utilisateurs de Gremlin doivent être en mesure de transmettre des valeurs de chaîne vides ("") en tant que propriétés de nœud et d'arête lors du chargement

de données CSV. Si la valeur `allowEmptyStrings` est définie sur `false` (valeur par défaut), ces chaînes vides sont traitées comme des valeurs nulles et ne sont pas chargées.

Si la valeur `allowEmptyStrings` est définie sur `true`, le chargeur traite les chaînes vides comme des valeurs de propriété valides et les charge en conséquence.

Pour de plus amples informations, veuillez consulter Graphe par défaut SPARQL et graphes nommés.

- **`updateSingleCardinalityProperties`** : paramètre facultatif qui contrôle la façon dont le chargeur en bloc traite une nouvelle valeur pour les propriétés de sommet ou d'arête à cardinalité unique.

  Valeurs autorisées : `"TRUE"`, `"FALSE"`.

  Valeur par défaut : `"FALSE"`.

  Par défaut, ou lorsque `updateSingleCardinalityProperties` est explicitement défini sur `"FALSE"`, le chargeur traite une nouvelle valeur comme une erreur, car elle ne respecte pas la cardinalité unique.

  En revanche, lorsque `updateSingleCardinalityProperties` est défini sur `"TRUE"`, le chargeur en bloc remplace la valeur existante par la nouvelle. Si plusieurs valeurs de propriété de sommet à cardinalité unique ou d'arête sont fournies dans le ou les fichiers source en cours de chargement, la valeur finale à l'issue du chargement en bloc peut être une de ces nouvelles valeurs. Le chargeur garantit uniquement que la valeur existante a été remplacée par une des nouvelles.

- **`queueRequest`** : paramètre d'indicateur facultatif qui indique si la demande de chargement peut être mise en file d'attente ou non.

Vous n'avez pas besoin d'attendre qu'une tâche de chargement soit terminée avant d'émettre la suivante, car Neptune peut mettre en file d'attente jusqu'à 64 tâches à la fois, si leurs paramètres `queueRequest` sont tous définis sur `"TRUE"`. L'ordre des tâches en file d'attente repose sur une structure FIFO (premier entré, premier sorti).

Si le paramètre `queueRequest` est omis ou défini sur `"FALSE"`, la demande de chargement échouera si une autre tâche de chargement est déjà en cours d'exécution.

Valeurs autorisées : `"TRUE"`, `"FALSE"`.

Valeur par défaut : `"FALSE"`.

- **`dependencies`** : paramètre facultatif qui peut subordonner une demande de chargement en file d'attente à la réussite d'une ou de plusieurs tâches précédentes dans la file d'attente.

  Neptune peut mettre en file d'attente jusqu'à 64 requêtes de chargement à la fois, si leurs paramètres `queueRequest` sont définis sur `"TRUE"`. Le paramètre `dependencies` vous permet de rendre l'exécution d'une telle requête en file d'attente dépendante de la réussite d'une ou de plusieurs requêtes précédentes spécifiées dans la file d'attente.

  Par exemple, si les charges `Job-A` et `Job-B` sont indépendantes l'une de l'autre, mais que la charge `Job-C` a besoin que `Job-A` et `Job-B` soient terminées avant de commencer, procédez comme suit :

  1. Soumettez `load-job-A` et `load-job-B` l'une après l'autre dans n'importe quel ordre, et enregistrez leurs ID de chargement.
  2. Soumettez `load-job-C` avec les ID de chargement des deux tâches dans son domaine `dependencies` :

  ```
  "dependencies" : ["job_A_load_id", "job_B_load_id"]
  ```

  En raison du paramètre `dependencies`, le chargeur en bloc ne démarrera pas `Job-C` avant que `Job-A` et `Job-B` soient terminées avec succès. Si l'un d'entre eux échoue, il ne Job-C sera pas exécuté et son statut sera défini sur`LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

  Vous pouvez configurer plusieurs niveaux de dépendance de cette façon, de sorte que l'échec d'une tâche entraîne l'annulation de toutes les demandes qui en dépendent directement ou indirectement.

- **`userProvidedEdgeIds`** : ce paramètre n'est obligatoire que lors du chargement de données openCypher contenant des ID de relation. Il doit être inclus et défini sur `True` lorsque les ID de relation openCypher sont explicitement fournis dans les données de chargement (recommandé).

  Quand `userProvidedEdgeIds` est absent ou défini sur `True`, une colonne `:ID` doit être présente dans chaque fichier de relations inclus dans le chargement.

  Quand `userProvidedEdgeIds` est présent et défini sur `False`, les fichiers de relations inclus dans le chargement ne doivent pas contenir de colonne `:ID`. Au lieu de cela, le chargeur Neptune génère automatiquement un ID pour chaque relation.

  Il est utile de fournir des ID de relation de manière explicite afin que le chargeur puisse reprendre le chargement après correction d'une erreur dans les données CSV, sans avoir à recharger les relations déjà chargées. Si aucun ID de relation n'a été attribué explicitement, le chargeur ne peut pas reprendre un chargement défaillant si un fichier de relation a dû être corrigé, et doit à la place recharger toutes les relations.
- `accessKey` : [obsolète] ID de clé d'accès d'un rôle IAM avec un accès au compartiment S3 et aux fichiers de données.

  Le paramètre `iamRoleArn` est recommandé à la place. Pour plus d'informations sur la création d'un rôle ayant accès à Amazon S3 et sur son association à un cluster Neptune, consultez Prérequis : rôle IAM et accès à Amazon S3.

  Pour plus d'informations, consultez Clés d'accès (ID de clé d'accès et clé d'accès secrète).
- `secretKey` : [obsolète] le paramètre `iamRoleArn` est recommandé à la place. Pour plus d'informations sur la création d'un rôle ayant accès à Amazon S3 et sur son association à un cluster Neptune, consultez Prérequis : rôle IAM et accès à Amazon S3.

  Pour plus d'informations, consultez Clés d'accès (ID de clé d'accès et clé d'accès secrète).

Considérations spéciales relatives au chargement de données openCypher

- Lorsque vous chargez des données openCypher au format CSV, le paramètre de format doit être défini sur `opencypher`.
- Le `updateSingleCardinalityProperties` paramètre est pris en charge pour les chargements OpenCypher. Toutes les propriétés OpenCypher ont une seule cardinalité, et si une valeur d'ID apparaît plusieurs fois, elle est traitée comme un doublon ou comme une erreur d'insertion (voir ci-dessous).

- Le chargeur Neptune gère les doublons qu'il trouve dans les données openCypher de la manière suivante :
  - Si le chargeur identifie plusieurs lignes avec le même ID de nœud, elles sont fusionnées selon la règle suivante :
    - Toutes les étiquettes des lignes sont ajoutées au nœud.
    - Pour chaque propriété, une seule des valeurs de propriété est chargée. Le choix de celle à charger n'est pas déterministe.
  - Si le chargeur identifie plusieurs lignes avec le même ID de relation, une seule d'entre elles est chargée. Le choix de celle à charger n'est pas déterministe.
  - Par défaut, le chargeur ne met pas à jour les valeurs des propriétés d'un nœud ou d'une relation existant dans la base de données s'il rencontre des données de charge ayant l'ID du nœud ou de la relation existant. Cependant, il charge les étiquettes et les propriétés des nœuds qui ne se trouvent pas dans le nœud ou la relation existants. Pour mettre à jour les valeurs de propriétés existantes, définissez `updateSingleCardinalityProperties` sur`"TRUE"`.
- Bien qu'il ne soit pas nécessaire d'attribuer des ID aux relations, cette approche est généralement conseillée (voir le paramètre `userProvidedEdgeIds` ci-dessus). Sans ID de relation explicites, le chargeur doit charger à nouveau toutes les relations en cas d'erreur dans un fichier de relations, plutôt que de reprendre le chargement là où il a échoué.

  De plus, si les données de chargement ne contiennent pas d'ID de relation explicites, le chargeur n'a aucun moyen de détecter les relations dupliquées.

Voici un exemple de commande de chargement openCypher :

AWS CLI

```
aws neptunedata start-loader-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --source "s3://bucket-name/object-key-name" \
  --format "opencypher" \
  --user-provided-edge-ids \
  --iam-role-arn "arn:aws:iam::account-id:role/role-name" \
  --s3-bucket-region "region" \
  --no-fail-on-error \
  --parallelism "MEDIUM"
```

Pour plus d'informations, consultez start-loader-job dans la AWS CLI référence des commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_loader_job(
    source='s3://bucket-name/object-key-name',
    format='opencypher',
    userProvidedEdgeIds=True,
    iamRoleArn='arn:aws:iam::account-id:role/role-name',
    s3BucketRegion='region',
    failOnError=False,
    parallelism='MEDIUM'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/loader \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://bucket-name/object-key-name",
        "format" : "opencypher",
        "userProvidedEdgeIds": "TRUE",
        "iamRoleArn" : "arn:aws:iam::account-id:role/role-name",
        "region" : "region",
        "failOnError" : "FALSE",
        "parallelism" : "MEDIUM"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -X POST https://your-neptune-endpoint:port/loader \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://bucket-name/object-key-name",
        "format" : "opencypher",
        "userProvidedEdgeIds": "TRUE",
        "iamRoleArn" : "arn:aws:iam::account-id:role/role-name",
        "region" : "region",
        "failOnError" : "FALSE",
        "parallelism" : "MEDIUM"
      }'
```

La réponse du chargeur est la même que d'habitude. Par exemple :

```
{
  "status" : "200 OK",
  "payload" : {
    "loadId" : "guid_as_string"
  }
}
```

Syntaxe de la réponse du chargeur Neptune

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "guid_as_string"
    }
}
```

200 OK

Une tâche de chargement démarrée avec succès renvoie un code `200`.

Erreurs du chargeur Neptune

Lorsqu'une erreur se produit, un objet JSON est renvoyé dans le `BODY` (corps) de la réponse. L'objet `message` contient une description de l'erreur.

Catégories d'erreurs

- `Error 400` : les erreurs de syntaxe renvoient une erreur de demande incorrecte `400` HTTP. Le message décrit l'erreur.
- `Error 500` : une demande valide qui ne peut pas être traitée renvoie une erreur de serveur interne HTTP `500`. Le message décrit l'erreur.

Voici les messages d'erreur possibles du chargeur avec une description de l'erreur.

Messages d'erreur du chargeur

- `Couldn't find the AWS credential for iam_role_arn` (HTTP 400)

  Les informations d'identification n'ont pas été trouvées. Vérifiez les informations d'identification fournies par rapport à la console ou à la AWS CLI sortie IAM. Vérifiez que vous avez ajouté au cluster le rôle IAM spécifié dans `iamRoleArn`.
- `S3 bucket not found for source` (HTTP 400)

  Le compartiment S3 n'existe pas. Vérifiez le nom du compartiment.
- `The source` *`source-uri`* `does not exist/not reachable` (HTTP 400)

  Aucun fichier correspondant n'a été trouvé dans le compartiment S3.
- `Unable to connect to S3 endpoint. Provided source =` *`source-uri`* `and region =` *`aws-region`* (HTTP 500)

  Impossible de se connecter à Amazon S3. La région doit correspondre à la région du cluster. Assurez-vous de disposer d'un point de terminaison de VPC. Pour plus d'informations sur la création d'un point de terminaison VPC, consultez Création d'un point de terminaison de VPC Amazon S3.
- `Bucket is not in provided Region (`*`aws-region`*`)` (HTTP 400)

Le bucket doit se trouver dans la même AWS région que votre instance de base de données Neptune.

- `Unable to perform S3 list operation` (HTTP 400)

  L'utilisateur ou le rôle IAM fourni ne dispose pas d'autorisations `List` sur le compartiment ou le dossier. Vérifiez la stratégie ou la liste de contrôle d'accès (ACL) sur le compartiment.

- `Start new load operation not permitted on a read replica instance` (HTTP 405)

  Le chargement est une opération d'écriture. Réessayez de charger sur le point de terminaison du read/write cluster.

- `Failed to start load because of unknown error from S3` (HTTP 500)

  Amazon S3 a renvoyé une erreur inconnue. Contactez AWS Support.

- `Invalid S3 access key` (HTTP 400)

  La clé d'accès n'est pas valide. Vérifiez les informations d'identification fournies.

- `Invalid S3 secret key` (HTTP 400)

  La clé secrète n'est pas valide. Vérifiez les informations d'identification fournies.

- `Max concurrent load limit breached` (HTTP 400)

  Si une demande de chargement est soumise sans `"queueRequest" : "TRUE"`, et qu'une tâche de chargement est en cours d'exécution, la demande échouera avec cette erreur.

- `Failed to start new load for the source "source name". Max load task queue size limit breached. Limit is 64` (HTTP 400)

  Neptune prend en charge la mise en file d'attente de jusqu'à 64 tâches de chargeur à la fois. Si une demande de chargement supplémentaire est soumise à la file d'attente alors qu'elle contient déjà 64 tâches, la demande échoue avec ce message.

Exemples de chargeur Neptune

Cet exemple montre comment utiliser le chargeur Neptune pour charger des données dans une base de données de graphes Neptune à l'aide du format CSV Gremlin. La demande est envoyée sous forme de requête HTTP POST au point de terminaison du chargeur Neptune, et le corps de la demande contient les paramètres nécessaires pour spécifier la source de données, le format, le rôle

IAM et les autres options de configuration. La réponse inclut l'ID de chargement, qui peut être utilisé pour suivre la progression du processus de chargement des données.

Example Demande

Voici une demande envoyée via HTTP POST à l'aide de la commande `curl`. Un fichier au format CSV Neptune est chargé. Pour de plus amples informations, veuillez consulter Format de chargement de données Gremlin.

AWS CLI

```
aws neptunedata start-loader-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --source "s3://bucket-name/object-key-name" \
  --format "csv" \
  --iam-role-arn "ARN for the IAM role you are using" \
  --s3-bucket-region "region" \
  --no-fail-on-error \
  --parallelism "MEDIUM" \
  --no-update-single-cardinality-properties \
  --no-queue-request
```

Pour plus d'informations, consultez start-loader-job dans la AWS CLI référence des commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_loader_job(
    source='s3://bucket-name/object-key-name',
    format='csv',
    iamRoleArn='ARN for the IAM role you are using',
    s3BucketRegion='region',
    failOnError=False,
    parallelism='MEDIUM',
    updateSingleCardinalityProperties=False,
```

```
    queueRequest=False
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/loader \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://bucket-name/object-key-name",
        "format" : "csv",
        "iamRoleArn" : "ARN for the IAM role you are using",
        "region" : "region",
        "failOnError" : "FALSE",
        "parallelism" : "MEDIUM",
        "updateSingleCardinalityProperties" : "FALSE",
        "queueRequest" : "FALSE"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -X POST https://your-neptune-endpoint:port/loader \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://bucket-name/object-key-name",
        "format" : "csv",
        "iamRoleArn" : "ARN for the IAM role you are using",
        "region" : "region",
        "failOnError" : "FALSE",
        "parallelism" : "MEDIUM",
        "updateSingleCardinalityProperties" : "FALSE",
        "queueRequest" : "FALSE"
```

```
        }'
```

Example Réponse

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

API Neptune Loader Get-Status

Obtient le statut d'une tâche `loader`.

Pour obtenir le statut de chargement, vous devez envoyer une demande `GET` HTTP au point de terminaison `https://`*`your-neptune-endpoint`*`:`*`port`*`/loader`. Pour obtenir le statut d'une demande de chargement particulière, vous devez inclure l'`loadId` comme paramètre d'URL ou ajouter l'`loadId` au chemin d'URL.

Neptune ne conserve une trace que des 1 024 tâches de chargement en bloc les plus récentes et ne stocke que les 10 000 dernières informations détaillées d'erreur par tâche.

Consultez Message d'erreur et de flux liés au chargeur Neptune pour obtenir la liste des messages d'erreur et de flux renvoyés par le chargeur en cas d'erreur.

Table des matières

- Demandes de Neptune Loader Get-Status
  - Syntaxe de Get-Status demande du chargeur
  - Paramètres de demande du Neptune Loader Get-Status
- Réponses du Neptune Loader Get-Status
  - Disposition JSON du Neptune Loader Get-Status Response
  - Objets de réponse Neptune Loader Get-Status OverallStatus et FailedFeeds
  - Objet de réponse aux Get-Status erreurs de Neptune Loader
  - Objet de réponse Get-Status ErrorLogs de Neptune Loader
- Exemples de chargeurs Neptune Get-Status

- Exemple de demande de statut de chargement
- Exemple de demande d'ID de chargement
- Exemple de demande de statut de détaillé
- Exemples de journaux d'erreurs du Neptune Loader Get-Status
  - Exemple de réponse de statut détaillée en cas d'erreur
  - Exemple d'erreur d'interruption d'une tâche de prélecture des données

Demandes de Neptune Loader Get-Status

Syntaxe de Get-Status demande du chargeur

```
GET https://your-neptune-endpoint:port/loader?loadId=loadId
```

```
GET https://your-neptune-endpoint:port/loader/loadId
```

```
GET https://your-neptune-endpoint:port/loader
```

Paramètres de demande du Neptune Loader Get-Status

- **`loadId`** : ID de la tâche de chargement. Si vous ne spécifiez pas d'ID `loadId`, une liste d'ID de chargement est renvoyée.
- **`details`** : inclut des détails autres que le statut global.

  Valeurs autorisées : `TRUE`, `FALSE`.

  Valeur par défaut : `FALSE`.
- **`errors`** : inclut la liste d'erreurs.

  Valeurs autorisées : `TRUE`, `FALSE`.

  Valeur par défaut : `FALSE`.

  La liste d'erreurs est paginée. Les paramètres `page` et `errorsPerPage` vous permettent de parcourir toutes les erreurs.
- **`page`** : numéro de page de l'erreur. Valide uniquement si le paramètre `errors` est défini sur `TRUE`.

  Valeurs autorisées : Entiers positifs

Valeur par défaut : 1.

- **`errorsPerPage`** : nombre d'erreurs par page. Valide uniquement si le paramètre `errors` est défini sur `TRUE`.

  Valeurs autorisées : Entiers positifs

  Valeur par défaut : 10.
- **`limit`** : nombre d'ID de chargement à répertorier. Valide uniquement lorsque vous demandez une liste d'ID de chargement en envoyant une demande `GET` sans `loadId` spécifié.

  Valeurs autorisées : nombre entier positif compris entre 1 et 100.

  Valeur par défaut : 100.
- **`includeQueuedLoads`** : paramètre facultatif qui peut être utilisé pour exclure les ID de chargement des demandes de chargement en file d'attente lorsqu'une liste d'ID de chargement est demandée.

  Par défaut, les ID de chargement de toutes les tâches de chargement avec le statut `LOAD_IN_QUEUE` sont inclus dans cette liste. Ils apparaissent avant les ID de chargement d'autres tâches, triés en fonction du moment où ils ont été ajoutés à la file d'attente, du plus récent au plus ancien.

  Valeurs autorisées : `TRUE`, `FALSE`.

  Valeur par défaut : `TRUE`.

Réponses du Neptune Loader Get-Status

L'exemple de réponse suivant de l' Get-Status API Neptune décrit la structure globale de la réponse, explique les différents champs et leurs types de données, ainsi que le traitement des erreurs et les détails du journal des erreurs.

Disposition JSON du Neptune Loader Get-Status Response

La structure générale d'une réponse de statut du chargeur est la suivante :

```
{
    "status" : "200 OK",
    "payload" : {
```

```
        "feedCount" : [
            {
                "LOAD_FAILED" : number
            }
        ],
        "overallStatus" : {
            "fullUri" : "s3://bucket/key",
            "runNumber" : number,
            "retryNumber" : number,
            "status" : "string",
            "totalTimeSpent" : number,
            "startTime" : number,
            "totalRecords" : number,
            "totalDuplicates" : number,
            "parsingErrors" : number,
            "datatypeMismatchErrors" : number,
            "insertErrors" : number,
        },
        "failedFeeds" : [
            {
                "fullUri" : "s3://bucket/key",
                "runNumber" : number,
                "retryNumber" : number,
                "status" : "string",
                "totalTimeSpent" : number,
                "startTime" : number,
                "totalRecords" : number,
                "totalDuplicates" : number,
                "parsingErrors" : number,
                "datatypeMismatchErrors" : number,
                "insertErrors" : number,
            }
        ],
        "errors" : {
            "startIndex" : number,
            "endIndex" : number,
            "loadId" : "string,
            "errorLogs" : [ ]
        }
    }
}
```

Objets de réponse Neptune Loader Get-Status OverallStatus et FailedFeeds

Les réponses possibles renvoyées pour chaque flux ayant échoué, avec une description de l'erreur, sont les mêmes que pour l'objet `overallStatus` dans une réponse `Get-Status`.

Ces champs apparaissent dans l'objet `overallStatus` pour tous les chargements, et le champ `failedFeeds` pour chaque flux ayant échoué.

- **`fullUri`** : URI du ou des fichiers à charger.

  Type : chaîne

  Format : `s3://`*`bucket`*`/`*`key`*.
- **`runNumber`** : nombre d'exécutions de ce chargement ou flux. Ce nombre est incrémenté lorsque le chargement est redémarré.

  Type : long non signé.
- **`retryNumber`** : nombre de nouvelles tentatives de ce chargement ou flux. Ce nombre est incrémenté lorsque le chargeur relance automatiquement un flux ou un chargement.

  Type : long non signé.
- **`status`** : statut renvoyé du chargement ou du flux. `LOAD_COMPLETED` indique la réussite du chargement, sans problèmes. Pour obtenir la liste des autres messages relatifs au statut du chargement, consultez Message d'erreur et de flux liés au chargeur Neptune.

  Type : chaîne
- **`totalTimeSpent`** : temps, en secondes, consacré à analyser et insérer les données pour le chargement ou le flux. N'inclut pas le temps consacré à l'extraction de la liste de fichiers source.

  Type : long non signé.
- **`totalRecords`** : nombre total d'enregistrements chargés ou de tentatives de chargement.

  Type : long non signé.

  Notez que lors du chargement à partir d'un fichier CSV, le nombre d'enregistrements ne fait pas référence au nombre de lignes chargées, mais plutôt au nombre d'enregistrements individuels contenus dans ces lignes. Par exemple, prenons un petit fichier CSV comme celui-ci :

```
~id,~label,name,team
```

```
'P-1','Player','Stokes','England'
```

Neptune considérerait que ce fichier contient trois enregistrements, à savoir :

```
P-1  label Player
P-1  name  Stokes
P-1  team  England
```

- **`totalDuplicates`** : nombre d'enregistrements en double détectés.

  Type : long non signé.

  Comme dans le cas du décompte `totalRecords`, cette valeur contient le nombre d'enregistrements en double individuels dans un fichier CSV, et non le nombre de lignes dupliquées. Prenons, par exemple, ce petit fichier CSV :

```
~id,~label,name,team
P-2,Player,Kohli,India
P-2,Player,Kohli,India
```

  Le statut renvoyé après le chargement ressemblerait à ceci, indiquant six enregistrements au total, dont trois sont des doublons :

```
{
  "status": "200 OK",
  "payload": {
    "feedCount": [
      {
        "LOAD_COMPLETED": 1
      }
    ],
    "overallStatus": {
      "fullUri": "(the URI of the CSV file)",
      "runNumber": 1,
      "retryNumber": 0,
      "status": "LOAD_COMPLETED",
      "totalTimeSpent": 3,
      "startTime": 1662131463,
      "totalRecords": 6,
      "totalDuplicates": 3,
      "parsingErrors": 0,
      "datatypeMismatchErrors": 0,
```

```
      "insertErrors": 0
    }
  }
}
```

Pour les chargements openCypher, un doublon est compté dans les situations suivantes :

- Le chargeur détecte qu'une ligne d'un fichier de nœud possède un ID sans espace d'ID identique à une autre valeur d'ID sans espace d'ID, que ce soit sur une autre ligne ou appartenant à un nœud existant.
- Le chargeur détecte qu'une ligne d'un fichier de nœud possède un ID avec un espace d'ID identique à une autre valeur d'ID avec un espace d'ID, que ce soit sur une autre ligne ou appartenant à un nœud existant.

Consultez Considérations spéciales relatives au chargement de données openCypher.

- **`parsingErrors`** : nombre d'erreurs d'analyse détectées.

  Type : long non signé.

- **`datatypeMismatchErrors`** : nombre d'enregistrements avec un type de données qui ne correspond pas aux données fournies.

  Type : long non signé.

- **`insertErrors`** : nombre d'enregistrements qui n'ont pas pu être insérés en raison d'erreurs.

  Type : long non signé.

Objet de réponse aux Get-Status **`erreurs`** de Neptune Loader

Les erreurs sont réparties dans les catégories suivantes :

- **`Error 400`** : une erreur non valide `loadId` renvoie une erreur de requête HTTP `400` incorrecte. Le message décrit l'erreur.
- **`Error 500`** : une demande valide qui ne peut pas être traitée renvoie une erreur de serveur interne HTTP `500`. Le message décrit l'erreur.

Consultez Message d'erreur et de flux liés au chargeur Neptune pour obtenir la liste des messages d'erreur et de flux renvoyés par le chargeur en cas d'erreur.

Lorsqu'une erreur se produit, un objet JSON `errors` est renvoyé dans le corps (BODY) de la réponse.

- **`startIndex`** : index de la première erreur incluse.

  Type : long non signé.
- **`endIndex`** : index de la dernière erreur incluse.

  Type : long non signé.
- **`loadId`** : ID du chargement. Vous pouvez utiliser cet ID pour afficher les erreurs du chargement en définissant le paramètre `errors` sur `TRUE`.

  Type : chaîne
- **`errorLogs`** : liste des erreurs.

  Type : liste.

Objet de réponse Get-Status **`ErrorLogs`** de Neptune Loader

L'`errorLogs`objet ci-dessous `errors` dans la Get-Status réponse du chargeur contient un objet décrivant chaque erreur à l'aide des champs suivants :

- **`errorCode`** : identifie la nature de l'erreur.

  Il peut avoir l'une des valeurs suivantes :
  - `PARSING_ERROR`
  - `S3_ACCESS_DENIED_ERROR`
  - `FROM_OR_TO_VERTEX_ARE_MISSING`
  - `ID_ASSIGNED_TO_MULTIPLE_EDGES`
  - `SINGLE_CARDINALITY_VIOLATION`
  - `FILE_MODIFICATION_OR_DELETION_ERROR`
  - `OUT_OF_MEMORY_ERROR`
  - `INTERNAL_ERROR` (renvoyé lorsque le chargeur en bloc ne parvient pas à déterminer le type d'erreur).
- **`errorMessage`** : message décrivant l'erreur.

Il peut s'agir d'un message générique associé au code d'erreur ou d'un message spécifique contenant des détails, par exemple sur un from/to sommet manquant ou sur une erreur d'analyse.

- **`fileName`** : nom du flux.
- **`recordNum`** : dans le cas d'une erreur d'analyse, il s'agit du numéro dans le fichier de l'enregistrement qui n'a pas pu être analysé. Il correspond à zéro si le numéro d'enregistrement n'est pas applicable à l'erreur ou s'il n'a pas pu être déterminé.

Par exemple, le chargeur en bloc générerait une erreur d'analyse s'il détectait une ligne défectueuse telle que la suivante dans un fichier RDF `nquads` :

```
<http://base#subject> |http://base#predicate> <http://base#true> .
```

Comme vous pouvez le constater, le deuxième `http` de la ligne ci-dessus devrait être précédé de < au lieu de |. L'objet d'erreur généré sous `errorLogs` dans une réponse de statut ressemble à ceci :

```
{
    "errorCode" : "PARSING_ERROR",
    "errorMessage" : "Expected '<', found: |",
    "fileName" : "s3://bucket/key",
    "recordNum" : 12345
},
```

Exemples de chargeurs Neptune Get-Status

Les exemples suivants illustrent l'utilisation de l' GET-Status API du chargeur Neptune, qui vous permet de récupérer des informations sur l'état de vos chargements de données dans la base de données de graphes Amazon Neptune. Ces exemples couvrent trois scénarios principaux : récupérer l'état d'une charge spécifique, répertorier les identifiants de charge disponibles et demander des informations détaillées sur l'état d'une charge spécifique.

Exemple de demande de statut de chargement

Voici une demande envoyée via HTTP `GET` à l'aide de la commande `curl`.

AWS CLI

```
aws neptunedata get-loader-job-status \
  --endpoint-url https://your-neptune-endpoint:port \
```

```
  --load-id loadId (a UUID)
```

Pour plus d'informations, consultez get-loader-job-status dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.get_loader_job_status(
    loadId='loadId (a UUID)'
)

print(response)
```

awscurl

```
awscurl 'https://your-neptune-endpoint:port/loader/loadId (a UUID)' \
  --region us-east-1 \
  --service neptune-db
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)'
```

Example Réponse

```
{
    "status" : "200 OK",
    "payload" : {
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        }
    }
}
```

Exemple de demande d'ID de chargement

Voici une demande envoyée via HTTP `GET` à l'aide de la commande `curl`.

AWS CLI

```
aws neptunedata list-loader-jobs \
  --endpoint-url https://your-neptune-endpoint:port \
  --limit 3
```

Pour plus d'informations, consultez list-loader-jobs dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
```

```
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.list_loader_jobs(
    limit=3
)

print(response)
```

awscurl

```
awscurl 'https://your-neptune-endpoint:port/loader?limit=3' \
  --region us-east-1 \
  --service neptune-db
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -X GET 'https://your-neptune-endpoint:port/loader?limit=3'
```

Example Réponse

```
{
    "status" : "200 OK",
    "payload" : {
         "loadIds" : [
            "a2c0ce44-a44b-4517-8cd4-1dc144a8e5b5",
            "09683a01-6f37-4774-bb1b-5620d87f1931",
            "58085eb8-ceb4-4029-a3dc-3840969826b9"
        ]
    }
}
```

Exemple de demande de statut de détaillé

Voici une demande envoyée via HTTP `GET` à l'aide de la commande `curl`.

AWS CLI

```
aws neptunedata get-loader-job-status \
  --endpoint-url https://your-neptune-endpoint:port \
  --load-id loadId (a UUID) \
  --details
```

Pour plus d'informations, consultez get-loader-job-status dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.get_loader_job_status(
    loadId='loadId (a UUID)',
    details=True
)

print(response)
```

awscurl

```
awscurl 'https://your-neptune-endpoint:port/loader/loadId (a UUID)?details=true' \
  --region us-east-1 \
  --service neptune-db
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)?details=true'
```

Example Réponse


```
{
    "status" : "200 OK",
    "payload" : {
        "failedFeeds" : [
            {
                "datatypeMismatchErrors" : 0,
                "fullUri" : "s3://bucket/key",
                "insertErrors" : 0,
                "parsingErrors" : 5,
                "retryNumber" : 0,
                "runNumber" : 1,
                "status" : "LOAD_FAILED",
                "totalDuplicates" : 0,
                "totalRecords" : 5,
                "totalTimeSpent" : 3.0
            }
        ],
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
```

```
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        }
    }
}
```

Exemples de journaux d'erreurs du Neptune Loader Get-Status

Les exemples suivants présentent la réponse d'état détaillée du chargeur Neptune lorsque des erreurs se sont produites pendant le processus de chargement des données. Les exemples illustrent la structure de la réponse, notamment les informations sur les flux défaillants, l'état général et les journaux d'erreurs détaillés.

Exemple de réponse de statut détaillée en cas d'erreur

Il s'agit d'une demande envoyée via HTTP `GET` avec `curl` :

AWS CLI

```
aws neptunedata get-loader-job-status \
  --endpoint-url https://your-neptune-endpoint:port \
  --load-id 0a237328-afd5-4574-a0bc-c29ce5f54802 \
  --details \
  --errors \
  --errors-per-page 3 \
  --page 1
```

Pour plus d'informations, consultez get-loader-job-status dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)
```

```
response = client.get_loader_job_status(
    loadId='0a237328-afd5-4574-a0bc-c29ce5f54802',
    details=True,
    errors=True,
    errorsPerPage=3,
    page=1
)

print(response)
```

awscurl

```
awscurl 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-
c29ce5f54802?details=true&errors=true&page=1&errorsPerPage=3' \
  --region us-east-1 \
  --service neptune-db
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -X GET 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-
c29ce5f54802?details=true&errors=true&page=1&errorsPerPage=3'
```

Example Exemple d'une réponse détaillée en cas d'erreur

Voici un exemple de la réponse que vous pourriez obtenir à la suite de la requête ci-dessus, avec un objet `errorLogs` répertoriant les erreurs de chargement détectées :

```
{
    "status" : "200 OK",
    "payload" : {
        "failedFeeds" : [
            {
                "datatypeMismatchErrors" : 0,
                "fullUri" : "s3://bucket/key",
```

```
                "insertErrors" : 0,
                "parsingErrors" : 5,
                "retryNumber" : 0,
                "runNumber" : 1,
                "status" : "LOAD_FAILED",
                "totalDuplicates" : 0,
                "totalRecords" : 5,
                "totalTimeSpent" : 3.0
            }
        ],
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        },
        "errors" : {
            "endIndex" : 3,
            "errorLogs" : [
                {
                    "errorCode" : "PARSING_ERROR",
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 1
                },
                {
                    "errorCode" : "PARSING_ERROR",
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 2
                },
                {
                    "errorCode" : "PARSING_ERROR",
```

```
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 3
                }
            ],
            "loadId" : "0a237328-afd5-4574-a0bc-c29ce5f54802",
            "startIndex" : 1
        }
    }
}
```

Exemple d'erreur d'interruption d'une **`tâche de prélecture des données`**

Parfois, lorsque vous obtenez un statut `LOAD_FAILED`, puis demandez des informations plus détaillées, l'erreur renvoyée peut être `PARSING_ERROR` avec un message `Data prefetch task interrupted`, comme suit :

```
"errorLogs" : [
    {
        "errorCode" : "PARSING_ERROR",
        "errorMessage" : "Data prefetch task interrupted: Data prefetch task for 11467
 failed",
        "fileName" : "s3://amzn-s3-demo-bucket/some-source-file",
        "recordNum" : 0
    }
]
```

Cette erreur se produit en cas d'interruption temporaire du processus de chargement des données qui n'est généralement pas provoquée par votre demande ni vos données. Elle peut généralement être résolue simplement en réexécutant la demande de chargement par lot. Si vous utilisez les paramètres par défaut, à savoir `"mode":"AUTO"` et `"failOnError":"TRUE"`, le chargeur ignore les fichiers qu'il a déjà chargés avec succès et reprend le chargement des fichiers qu'il n'avait pas encore chargés lorsque l'interruption s'est produite.

Tâche d'annulation du chargeur Neptune

Annule une tâche de chargement.

Pour annuler une tâche, vous devez envoyer une demande `DELETE` HTTP au point de terminaison `https://your-neptune-endpoint:port/loader`. L'ID `loadId` peut être ajouté au chemin d'URL `/loader`, ou inclus en tant que variable dans l'URL.

Syntaxe de la demande d'annulation de tâche

```
DELETE https://your-neptune-endpoint:port/loader?loadId=loadId
```

```
DELETE https://your-neptune-endpoint:port/loader/loadId
```

Paramètres de demande d'annulation de tâche

loadId

ID de la tâche de chargement.

Syntaxe de la réponse d'annulation de tâche

```
no response body
```

200 OK

Une tâche de chargement supprimée avec succès renvoie un code `200`.

Erreurs d'annulation de tâche

Lorsqu'une erreur se produit, un objet JSON est renvoyé dans le `BODY` (corps) de la réponse. L'objet `message` contient une description de l'erreur.

Catégories d'erreurs

- **`Error 400`** : une erreur non valide `loadId` renvoie une erreur de requête HTTP `400` incorrecte. Le message décrit l'erreur.
- **`Error 500`** : une demande valide qui ne peut pas être traitée renvoie une erreur de serveur interne HTTP `500`. Le message décrit l'erreur.

Messages d'erreur d'annulation de tâche

Voici les messages d'erreur possibles de l'API d'annulation avec une description de l'erreur.

- `The load with id = load_id does not exist or not active` (HTTP 404) : le chargement est introuvable. Vérifiez la valeur du paramètre `id`.

- `Load cancellation is not permitted on a read replica instance.` (HTTP 405) : le chargement est une opération d'écriture. Réessayez de charger sur le point de terminaison du read/write cluster.

Exemples d'annulation de tâche

Example Demande

Voici une demande envoyée via HTTP `DELETE` à l'aide de la commande `curl`.

AWS CLI

```
aws neptunedata cancel-loader-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --load-id 0a237328-afd5-4574-a0bc-c29ce5f54802
```

Pour plus d'informations, consultez cancel-loader-job dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.cancel_loader_job(
    loadId='0a237328-afd5-4574-a0bc-c29ce5f54802'
)

print(response)
```

awscurl

```
awscurl 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-
c29ce5f54802' \
  --region us-east-1 \
  --service neptune-db \
```

```
  -X DELETE
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -X DELETE 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-c29ce5f54802'
```

Utilisation AWS Database Migration Service pour charger des données dans Amazon Neptune à partir d'un autre magasin de données

AWS Database Migration Service (AWS DMS) peut charger des données dans Neptune à partir de bases de données sources prises en charge rapidement et en toute sécurité. La base de données source reste pleinement opérationnelle durant la migration, ce qui réduit au minimum les temps d'arrêt des applications qui l'utilisent.

Vous trouverez des informations détaillées AWS DMS à ce sujet dans le guide de l'AWS Database Migration Service utilisateur et dans la référence de l'AWS Database Migration Service API. En particulier, vous pourrez découvrir comment configurer un cluster Neptune en tant que cible de la migration dans Utilisation d'Amazon Neptune comme cible pour AWS Database Migration Service.

Voici quelques prérequis à l'importation de données dans Neptune à l'aide de AWS DMS :

- Vous devrez créer un objet de mappage de AWS DMS table pour définir la manière dont les données doivent être extraites de la base de données source (voir Spécification de la sélection de tables et des transformations par mappage de table à l'aide de JSON dans le AWS DMS guide de l'utilisateur pour plus de détails). Cet objet de configuration de mappage de table spécifie les tables qui doivent être lues et l'ordre dans lequel cette lecture doit s'effectuer, ainsi que les noms des colonnes. Il peut également filtrer les lignes copiées et fournir des transformations de valeur simples telles que la conversion en minuscules ou l'arrondi.

- Vous devrez créer une configuration `GraphMappingConfig` Neptune pour spécifier la façon dont les données extraites de la base de données source doivent être chargées dans Neptune. Pour les données RDF (interrogées à l'aide de SPARQL), l'élément `GraphMappingConfig` est écrit dans le langage de mappage standard W3 R2RML. Pour les données de graphe de propriétés (interrogées à l'aide de Gremlin), `GraphMappingConfig` est un objet JSON, décrit dans GraphMappingConfig Mise en page des Property-Graph/Gremlin données.
- Vous devez l'utiliser AWS DMS pour créer une instance de réplication dans le même VPC que votre cluster de base de données Neptune, afin de faciliter le transfert de données.
- Vous aurez également besoin d'un compartiment Amazon S3 à utiliser comme stockage intermédiaire pour le transfert des données de migration.

Création d'un Neptune GraphMappingConfig

L'élément `GraphMappingConfig` que vous créez spécifie la façon dont les données extraites d'un magasin de données source doivent être chargées dans un cluster de bases de données Neptune. Son format diffère selon qu'il est destiné au chargement de données RDF ou au chargement de données de graphe de propriétés.

Pour les données RDF, vous pouvez utiliser le langage W3 R2RML pour mapper des données relationnelles sur RDF.

Si vous chargez des données de graphe de propriétés destinées à être interrogées à l'aide de Gremlin, vous créez un objet JSON pour `GraphMappingConfig`.

GraphMappingConfig Mise en page des RDF/SPARQL données

Si vous chargez des données RDF destinées à être interrogées à l'aide de SPARQL, vous écrivez l'élément `GraphMappingConfig` en R2RML. `R2RML` est un langage W3 standard destiné au mappage des données relationnelles sur RDF. Voici un exemple :

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/ns#> .

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "nodes" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{id}";
        rr:class ex:Employee;
    ];
```

```
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "label" ];
    ] .
```

Voici un autre exemple :

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<#TriplesMap2>
    rr:logicalTable [ rr:tableName "Student" ];
    rr:subjectMap   [ rr:template "http://example.com/{ID}{Name}";
                      rr:class foaf:Person ];
    rr:predicateObjectMap [
        rr:predicate ex:id ;
        rr:objectMap  [ rr:column "ID";
                        rr:datatype xsd:integer ]
    ];
    rr:predicateObjectMap [
        rr:predicate foaf:name ;
        rr:objectMap  [ rr:column "Name" ]
    ] .
```

La recommandation W3 dans R2RML: RDB to RDF Mapping Language fournit des détails sur le langage.

GraphMappingConfig Mise en page des Property-Graph/Gremlin données

L'élément `GraphMappingConfig` comparable pour les données de graphe de propriétés est un objet JSON qui fournit une règle de mappage pour chaque entité de graphe à générer à partir des données source. Le modèle suivant montre à quoi ressemble chaque règle de cet objet :

```
{
  "rules": [
    {
      "rule_id": "(an identifier for this rule)",
      "rule_name": "(a name for this rule)",
      "table_name": "(the name of the table or view being loaded)",
      "vertex_definitions": [
```

```
        {
          "vertex_id_template": "{col1}",
          "vertex_label": "(the vertex to create)",
          "vertex_definition_id": "(an identifier for this vertex)",
          "vertex_properties": [
            {
              "property_name": "(name of the property)",
              "property_value_template": "{col2} or text",
              "property_value_type": "(data type of the property)"
            }
          ]
        }
      ]
    },
    {
      "rule_id": "(an identifier for this rule)",
      "rule_name": "(a name for this rule)",
      "table_name": "(the name of the table or view being loaded)",
      "edge_definitions": [
        {
          "from_vertex": {
            "vertex_id_template": "{col1}",
            "vertex_definition_id": "(an identifier for the vertex referenced above)"
          },
          "to_vertex": {
            "vertex_id_template": "{col3}",
            "vertex_definition_id": "(an identifier for the vertex referenced above)"
          },
          "edge_id_template": {
            "label": "(the edge label to add)",
            "template": "{col1}_{col3}"
          },
          "edge_properties":[
            {
              "property_name": "(the property to add)",
              "property_value_template": "{col4} or text",
              "property_value_type": "(data type like String, int, double)"
            }
          ]
        }
      ]
    }
  ]
```

```
}
```

Notez que la présence d'une étiquette de sommet implique que le sommet est créé ici, alors que son absence implique que le sommet est créé par une source différente et que cette définition n'ajoute que des propriétés de sommet.

Voici un exemple de règle pour un enregistrement d'employé :

```
{
  "rules": [
    {
      "rule_id": "1",
      "rule_name": "vertex_mapping_rule_from_nodes",
      "table_name": "nodes",
      "vertex_definitions": [
        {
          "vertex_id_template": "{emp_id}",
          "vertex_label": "employee",
          "vertex_definition_id": "1",
          "vertex_properties": [
            {
              "property_name": "name",
              "property_value_template": "{emp_name}",
              "property_value_type": "String"
            }
          ]
        }
      ]
    },
    {
      "rule_id": "2",
      "rule_name": "edge_mapping_rule_from_emp",
      "table_name": "nodes",
      "edge_definitions": [
        {
          "from_vertex": {
            "vertex_id_template": "{emp_id}",
            "vertex_definition_id": "1"
          },
          "to_vertex": {
            "vertex_id_template": "{mgr_id}",
            "vertex_definition_id": "1"
          },
```

```
          "edge_id_template": {
            "label": "reportsTo",
            "template": "{emp_id}_{mgr_id}"
          },
          "edge_properties":[
            {
              "property_name": "team",
              "property_value_template": "{team}",
              "property_value_type": "String"
            }
          ]
        }
      ]
    }
  ]
}
```

Création d'un AWS DMS tâche de réplication avec Neptune comme cible

Une fois que vous avez créé vos configurations de mappage de tables et de mappage de graphes, procédez comme suit pour charger les données du magasin source dans Neptune. Consultez la AWS DMS documentation pour plus de détails sur les API en question.

Créez un AWS DMS instance de réplication

Créez une instance de AWS DMS réplication dans le VPC sur lequel votre cluster de base de données Neptune est exécuté (voir Utilisation d'une instance de réplication AWS DMS et CreateReplicationInstancedans le guide de l'utilisateur). AWS DMS Pour ce faire, vous pouvez utiliser une AWS CLI commande comme celle-ci :

```
aws dms create-replication-instance \
    --replication-instance-identifier (the replication instance identifier) \
    --replication-instance-class (the size and capacity of the instance, like
 'dms.t2.medium') \
    --allocated-storage (the number of gigabytes to allocate for the instance
 initially) \
    --engine-version (the DMS engine version that the instance should use) \
    --vpc-security-group-ids (the security group to be used with the instance)
```

Créez un AWS DMS point de terminaison pour la base de données source

L'étape suivante consiste à créer un AWS DMS point de terminaison pour votre magasin de données source. Vous pouvez utiliser l' AWS DMS CreateEndpointAPI de la AWS CLI manière suivante :

```
aws dms create-endpoint \
    --endpoint-identifier (source endpoint identifier) \
    --endpoint-type source \
    --engine-name (name of source database engine) \
    --username (user name for database login) \
    --password (password for login) \
    --server-name (name of the server) \
    --port (port number) \
    --database-name (database name)
```

Configuration d'un compartiment Amazon S3 que Neptune utilisera pour le transfert de données

Si vous ne disposez pas d'un compartiment Amazon S3 que vous pouvez utiliser pour transférer des données, créez-en un comme expliqué dans la section Création d'un compartiment dans le Getting-Started guide Amazon S3 ou Comment créer un compartiment S3 ? dans le guide de l'utilisateur de la console.

Vous devrez créer une politique IAM octroyant les autorisations `GetObject`, `PutObject`, `DeleteObject` et `ListObject` au compartiment si vous n'en avez pas déjà une :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket"
```

```
      ]
    },
    {
      "Sid": "AllObjectActions",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

Si l'authentification IAM est activée pour votre cluster de bases de données Neptune, vous devrez également inclure la politique suivante :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:111122223333:cluster-resource-id/
*"
    }
  ]
}
```

Créez un rôle IAM en tant que document d'approbation auquel attacher la politique :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "dms.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Sid": "neptune",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Après avoir attaché la politique au rôle, attachez-le à votre cluster de bases de données Neptune. Cela permettra d' AWS DMS utiliser le bucket pour préparer les données en cours de chargement.

Création d'un point de terminaison Amazon S3 dans le VPC Neptune

Créez maintenant un point de terminaison de passerelle de VPC pour votre compartiment Amazon S3 intermédiaire dans le VPC où se trouve votre cluster Neptune. Pour ce faire, vous pouvez AWS CLI utiliser le AWS Management Console ou le, comme décrit dans Création d'un point de terminaison de passerelle.

Créez un AWS DMS point de terminaison cible pour Neptune

Créez un AWS DMS point de terminaison pour votre cluster de base de données Neptune cible. Vous pouvez utiliser l' AWS DMS CreateEndpointAPI avec le `NeptuneSettings` paramètre suivant :

```
aws dms create-endpoint \
```

```
    --endpoint-identifier (target endpoint identifier) \
    --endpoint-type target \
    --engine-name neptune \
    --server-name (name of the server) \
    --port (port number) \
    --neptune-settings '{ \
      "ServiceAccessRoleArn": "(ARN of the service access role)", \
      "S3BucketName": "(name of S3 bucket to use for staging files when migrating)", \
      "S3BucketFolder": "(name of the folder to use in that S3 bucket)", \
      "ErrorRetryDuration": (number of milliseconds to wait between bulk-load retries),
\
      "MaxRetryCount": (the maximum number of times to retry a failing bulk-load job),
\
      "MaxFileSize": (maximum file size, in bytes, of the staging files written to S3),
\
      "IamAuthEnabled": (set to true if IAM authentication is enabled on the Neptune
cluster) }'
```

L'objet JSON transmis à l' AWS DMS `CreateEndpoint`API dans son `NeptuneSettings` paramètre comporte les champs suivants :

- **`ServiceAccessRoleArn`** : (obligatoire) ARN d'un rôle IAM qui permet un accès précis au compartiment S3 utilisé comme outil intermédiaire de la migration des données vers Neptune. Ce rôle doit également avoir des autorisations d'accès à votre cluster de bases de données Neptune si l'autorisation IAM y est activée.
- **`S3BucketName`** : (obligatoire) pour la migration à chargement complet, l'instance de réplication convertit toutes les données RDS en fichiers CSV ou quad et les charge vers ce compartiment intermédiaire dans S3, puis les charge en bloc dans Neptune.
- **`S3BucketFolder`** : (obligatoire) dossier à utiliser dans le compartiment intermédiaire S3.
- **`ErrorRetryDuration`** : (facultatif) nombre de millisecondes à attendre après l'échec d'une demande Neptune avant d'effectuer une nouvelle demande. La valeur par défaut est 250.
- **`MaxRetryCount`**— (facultatif) Le nombre maximum de demandes de nouvelle tentative à effectuer AWS DMS après un échec de nouvelle tentative. La valeur par défaut est 5.
- **`MaxFileSize`** : (facultatif) taille maximale en octets de chaque fichier intermédiaire enregistré dans S3 lors de la migration. La valeur par défaut est de 1 048 576 Ko (1 Go).
- **`IsIAMAuthEnabled`** : (facultatif) s'il est défini sur `true`, l'authentification IAM est activée sur le cluster de bases de données Neptune. Dans le cas contraire, sa valeur indique `false`. La valeur par défaut est `false`.

Testez les connexions aux nouveaux points de terminaison

Vous pouvez tester la connexion à chacun de ces nouveaux points de terminaison à l'aide de l' AWS DMS TestConnectionAPI comme suit :

```
aws dms test-connection \
    --replication-instance-arn (the ARN of the replication instance) \
    --endpoint-arn (the ARN of the endpoint you are testing)
```

Créez un AWS DMS tâche de réplication

Une fois les étapes précédentes terminées avec succès, créez une tâche de réplication pour migrer les données de votre banque de données source vers Neptune, à l'aide de AWS DMS CreateReplicationTaskl'API suivante :

```
aws dms create-replication-task \
    --replication-task-identifier (name for the replication task) \
    --source-endpoint-arn (ARN of the source endpoint) \
    --target-endpoint-arn (ARN of the target endpoint) \
    --replication-instance-arn (ARN of the replication instance) \
    --migration-type full-load \
    --table-mappings (table-mapping JSON object or URI like 'file:///tmp/table-
mappings,json') \
    --task-data (a GraphMappingConfig object or URI like 'file:///tmp/graph-mapping-
config.json')
```

Le paramètre `TaskData` fournit le GraphMappingConfig qui spécifie la façon dont les données en cours de copie doivent être stockées dans Neptune.

Démarrez le AWS DMS tâche de réplication

Vous pouvez maintenant démarrer la tâche de réplication :

```
aws dms start-replication-task
    --replication-task-arn (ARN of the replication task started in the previous step)
    --start-replication-task-type start-replication
```

Chargement de données dans Amazon Neptune à l'aide de requêtes

Neptune prend en charge l'écriture de données directement par le biais d'opérations en langage de requête. Vous pouvez utiliser des opérations d'écriture standard telles que `CREATE` et `MERGE` dans OpenCypher, `INSERT` dans SPARQL ou `mergeV()` `mergeE()` dans Gremlin pour ajouter ou modifier des données dans votre graphique. Ces opérations conviennent aux mises à jour incrémentielles et aux écritures transactionnelles.

Pour charger des données depuis Amazon S3, utilisez le Utilisation du chargeur en vrac Amazon Neptune pour ingérer des données pour les grands ensembles de données nécessitant des performances optimisées. Pour les petits ensembles de données contenus dans un ou plusieurs fichiers Amazon S3, vous pouvez utiliser des fonctions de chargement basées sur des requêtes pour lire et traiter les données directement dans vos requêtes.

Les fonctions de chargement basées sur des requêtes suivantes sont disponibles :

OpenCypher : neptune.read ()

La `neptune.read()` fonction lit les fichiers CSV ou Parquet depuis Amazon S3 dans une `CALL` sous-requête, ce qui vous permet de traiter et de charger des données au moment de la requête.

```
CALL neptune.read({
  source: "s3://bucket/data.csv",
  format: "csv"
})
YIELD row
CREATE (n:Person {id: row.id, name: row.name})
```

Pour une documentation complète, voirneptune.read ().

SPARQL : CHARGER et DÉCHARGER

Les `LOAD` opérations SPARQL importent des données RDF depuis un URI dans un graphe nommé. `UNLOAD`exporte les données d'un graphique vers Amazon S3.

```
LOAD <s3://bucket/data.ttl> INTO GRAPH <http://example.org/graph>
```

Pour une documentation complète, voirUtilisation de SPARQL UPDATE LOAD pour importer des données dans Neptune.

Gmail : io () étape

Vous pouvez également utiliser l'`g.io(URL).read()`étape de Gremlin pour lire des fichiers de données au format GraphML (format XML), GraphSon (format JSON) et dans d'autres formats.

```
g.io("s3://bucket/data.graphml").read().iterate()
```

Consultez TinkerPopla documentation pour plus de détails.

Stockage

Neptune prend en charge la collecte des déchets par dictionnaire (GC) pour les données des graphes de propriétés, qui peut être activée via le `neptune_lab_mode` paramètre lorsqu'il n'`neptune_streams`est pas actif. Lorsqu'elle est activée, cette tâche en arrière-plan nettoie les entrées de dictionnaire inutilisées, ce qui peut réduire le taux de croissance des données. La fonctionnalité peut fonctionner selon deux modes : soft_delete (marquer les entrées comme supprimées sans suppression explicite) et activé (suppression explicite des entrées). Le processus GC peut avoir un impact sur les performances du système en se heurtant à des fils de requête pour des ressources telles que le processeur et le cache tampon, et peut s'exécuter avec une simultanéité maximale de 16 threads.

Neptune prend également en charge les identifiants Edge générés par le serveur en ligne, qui peuvent être activés via un paramètre de configuration lorsque neptune_streams n'est pas actif. Lorsque cette fonctionnalité est activée, le serveur génère des identifiants intégrés uniques pour les périphériques qui n'ont pas d'identifiant défini par l'utilisateur, en utilisant le préfixe réservé « neptune_reserved ». Ces identifiants intégrés ne sont pas stockés dans le dictionnaire, ce qui peut améliorer l'efficacité du stockage.

Rubriques

- Collecte des déchets du dictionnaire Neptune
- Identifiant Edge intégré Neptune généré par le serveur

Collecte des déchets du dictionnaire Neptune

Neptune prend en charge la collecte des déchets par dictionnaire (GC), qui peut être activée via le `neptune_lab_mode` paramètre relatif aux données du graphe de propriétés. Il peut être activé pour les clusters contenant uniquement des données de graphes de propriétés lorsqu'il n'`neptune_streams`est pas activé. La fonctionnalité est automatiquement désactivée si elle `neptune_streams` est activée ou s'il existe des `neptune_streams` données non expirées. La fonctionnalité nécessite le redémarrage d'une instance d'écriture pour être activée. Cette fonctionnalité est disponible à partir de la version 1.4.3.0 du moteur.

Lorsque cette option est activée, les entrées de dictionnaire inutilisées sont nettoyées par une tâche en arrière-plan. Il ne réduit pas`VolumeBytesUsed`, mais libère de l'espace dans l'index pour les

nouvelles insertions. Le taux de croissance `VolumeBytesUsed` est susceptible d'être inférieur lorsque le dictionnaire GC est activé par rapport à lorsqu'il ne l'est pas.

La collecte des déchets du dictionnaire s'exécute en arrière-plan et analyse toutes les données des graphes et des dictionnaires pour trouver les termes qui ne sont pas utilisés. Une nouvelle exécution est déclenchée au démarrage dès qu'environ 6 % des données ont changé. GC est en concurrence avec les fils de requête pour les ressources du serveur telles que le processeur, le cache tampon, la génération de journaux d'annulation et les I/O opérations d'écriture, ce qui peut réduire le débit des requêtes. Comme GC analyse les données qui ne sont pas activement touchées par les requêtes, cela peut avoir un impact sur le cache tampon du nœud d'écriture. Le cluster peut subir des I/O opérations d'écriture supplémentaires et avoir davantage de journaux d'annulation à purger lorsque le GC effectue de nouvelles suppressions, ce qui peut également entraîner des valeurs plus élevées pour la `UndoLogListSize` métrique.

GC peut être exécuté en deux modes, `soft_delete` et`enabled`. Lorsqu'elles sont exécutées dans ce `soft_delete` mode, les entrées de dictionnaire non utilisées sont marquées comme supprimées (soft_delete) mais ne sont pas supprimées explicitement. Ce mode peut également être utilisé pour comprendre les caractéristiques de performance une fois le fonctionnement en arrière-plan activé. Lorsque le `enabled` mode est utilisé, les entrées sont explicitement supprimées (suppression « dure »). Il est recommandé d'exécuter GC en `soft_delete` mode pendant un certain temps avant de passer `enabled` en mode.

Dictionary GC prend en charge une simultanéité maximale de 16 (sur les machines dotées de 16 cœurs ou plus). Il s'exécute par défaut avec un seul thread, mais il peut être exécuté avec une plus grande simultanéité lorsqu'il est activé pour la première fois. Les threads GC du dictionnaire s'exécutent avec la même priorité que les fils de requête, et ils utilisent les ressources de l'auteur de la même manière.

Le dictionnaire GC peut être activé via le `neptune_lab_mode` paramètre en réglant la `DictionaryGCMode` clé. Il accepte trois valeurs possibles : `disabled` (par défaut)`soft_delete`, ou`enabled`. Par exemple, l'exemple de code suivant définirait la valeur `DictionaryGCMode` sur `soft_delete` :

```
neptune_lab_mode = 'DictionaryGCMode=soft_delete'
```

Le paramètre de simultanéité `DictionaryGCConcurrency` est facultatif et peut prendre une valeur comprise entre 1 et 16. S'il est défini sur une valeur supérieure au minimum de 16 et au nombre de cœurs, la simultanéité est plafonnée à cette valeur.

```
neptune_lab_mode = 'DictionaryGCMode=soft_delete,DictionaryGCConcurrency=2'
```

La tâche GC du dictionnaire est activée en arrière-plan après le démarrage du serveur, une fois que certaines données sont disponibles. L'état du moteur affiche l'état actuel du dictionnaire GC. L'exemple de sortie ci-dessous montre que le dictionnaire GC est en `soft_delete` mode et fonctionne avec une simultanéité de 2. Si la tâche en arrière-plan est en cours d'exécution, il peut s'agir de rechercher activement les entrées de dictionnaire inutilisées et de les supprimer, ou d'attendre une nouvelle série de suppressions pour déclencher un nouveau cycle de GC.

```
{"status":"healthy",...,"labMode":
{"ObjectIndex":"disabled","DictionaryGC":"{Mode=enabled,Concurrency=2}"},...}
```

Le dictionnaire GC est suspendu lorsque l'une des conditions suivantes est remplie :

- Chargement en vrac actif.
- La mémoire disponible est inférieure à 15 Go.
- `UndoLogListSize`est supérieur à 1 000 000.

Identifiant Edge intégré Neptune généré par le serveur

Neptune prend en charge les identifiants Edge intégrés. Server-Generated Il peut être activé via le paramètre de configuration Neptune `neptune_enable_inline_server_generated_edge_id` lorsqu'il n'`neptune_streams`est pas activé. Cette fonctionnalité est disponible pour les requêtes Gkremlin à partir de la version 1.4.3.0 du moteur, et sera disponible pour les OpenCypher requêtes dans une future version.

L'identifiant Edge est un identifiant unique pour un bord. Un identifiant de bord peut être fourni lors de l'insertion d'un bord. Si aucun identifiant n'est fourni, le serveur génère et attribue un identifiant basé sur l'UUID au périphérique par défaut. Tout comme l'identifiant défini par l'utilisateur, l'identifiant UUID-based généré par le serveur est stocké dans le dictionnaire.

Lorsque la `neptune_enable_inline_server_generated_edge_id` fonctionnalité est activée, le serveur génère un identifiant intégré unique lorsqu'aucun identifiant n'est fourni dans la requête. Les identifiants de bord intégrés ne sont pas stockés dans le dictionnaire, ce qui améliore l'efficacité du stockage. Les identifiants intégrés générés par le serveur commencent par le préfixe réservé. `neptune_reserved`

> **Warning**
>
> Neptune réserve le `'neptune_reserved'` préfixe pour les identifiants intégrés générés par le serveur. Une erreur s'affichera pour les requêtes qui tentent d'insérer des données avec un identifiant défini par l'utilisateur commençant par le préfixe réservé.

La fonctionnalité Edge ID générée par le serveur intégré peut être activée en définissant le paramètre au niveau du cluster sur. `neptune_enable_inline_server_generated_edge_id` 1 Un redémarrage de l'instance est requis. L'exemple suivant active la fonctionnalité Edge ID générée par le serveur :

```
"ParameterName=neptune_enable_inline_server_generated_edge_id,ParameterValue=1,ApplyMethod=pend
reboot"
```

Pour vérifier si la fonctionnalité est activée, vous pouvez vérifier les fonctionnalités dans l'état du moteur. Cette fonctionnalité est automatiquement désactivée si elle `neptune_streams` est activée. L'exemple de sortie suivant montre l'état du moteur pour la fonctionnalité activée :

```
"features":{"InlineServerGeneratedEdgeId":"enabled"}
```

L'exemple suivant de G705 ajoute un Edge sans ID défini par l'utilisateur lorsque la fonctionnalité Edge ID générée par le serveur en ligne est activée :

```
curl - X POST--url https: //<neptune-cluster-endpoint>:8182/gremlin/ --data
 '{"gremlin":"g.withSideEffect(\"Neptune#disablePushdownOptimization\",
 true).addV().property(id, \"a\").addV().property(id, \"b\").addE(\"el\").to(V(\"a
\"))"}'
{
    "requestId": "b6b84605-53ad-4c04-baf1-7f0f31a3aeaf",
    "status": {
        "message": "",
        "code": 200,
        "attributes": {
            "@type": "g:Map",
            "@value": []
        }
    },
    "result": {
        "data": {
```

```
            "@type": "g:List",
            "@value": [{
                "@type": "g:Edge",
                "@value": {
                    "id": "neptune_reserved_231850767",
                    "label": "el",
                    "inVLabel": "vertex",
                    "outVLabel": "vertex",
                    "inV": "a",
                    "outV": "b"
                }
            }]
        },
        "meta": {
            "@type": "g:Map",
            "@value": []
        }
    }
}
```

Interrogation d'un graphe Neptune

Neptune prend en charge deux langages d'interrogation pour accéder à un graphe :

- Gkremlin, défini par Apache TinkerPop pour créer et interroger des graphes de propriétés.

  Dans Gremlin, une requête est une traversée composée d'étapes distinctes, chacune suivant une arête jusqu'à un nœud.

  Consultez Accès au graphe Neptune avec Gremlin pour en savoir plus sur l'utilisation de Gremlin dans Neptune, et Conformité d'Amazon Neptune avec les normes Gremlin pour obtenir des informations spécifiques sur l'implémentation Neptune de Gremlin.
- openCypher est un langage de requête déclaratif pour les graphes de propriétés initialement développé par Neo4j, puis rendu open source en 2015. Il a contribué au projet openCypher sous une licence open source Apache 2. Sa syntaxe est documentée dans la spécification openCypher.
- SPARQL est un langage déclaratif basé sur la correspondance de modèles de graphes pour interroger des données RDF. Il est pris en charge par le World Wide Web Consortium.

  Consultez Accès au graphe Neptune avec SPARQL pour en savoir plus sur l'utilisation de SPARQL dans Neptune, et Conformité d'Amazon Neptune avec les normes SPARQL pour obtenir des informations spécifiques sur l'implémentation Neptune de SPARQL.

> **Note**
>
> Gremlin et openCypher peuvent tous deux être utilisés pour interroger toutes les données de graphes de propriétés stockées dans Neptune, quelle que soit la manière dont elles ont été chargées.

Rubriques

- Mise en file d'attente des requêtes dans Amazon Neptune
- Cache du plan de requête dans Amazon Neptune
- Injection d'un ID personnalisé dans une requête Neptune Gremlin ou SPARQL
- Accès au graphe Neptune avec Gremlin
- Accès au graphe Neptune avec openCypher

- Accès au graphe Neptune avec SPARQL

Mise en file d'attente des requêtes dans Amazon Neptune

Lors du développement et du réglage d'applications graphiques, il peut être utile de connaître les implications de la façon dont les requêtes sont mises en file d'attente par la base de données. Dans Amazon Neptune, la mise en file d'attente des requêtes présente les caractéristiques suivantes :

- Le nombre maximal de requêtes pouvant être mises en file d'attente par instance, quelle que soit la taille de l'instance, est de 8 192. Toutes les requêtes au-delà de ce nombre sont rejetées et échouent avec une exception `ThrottlingException`.
- Le nombre maximal de requêtes pouvant être exécutées en même temps est déterminé par le nombre de threads de travail affectés, qui est généralement défini par le double du nombre de cœurs de processeurs virtuels (vCPU) disponibles.
- La latence de requête inclut le temps passé par une requête dans la file d'attente, ainsi que le retour sur le réseau et le temps d'exécution réel.

Comment déterminer le nombre de requêtes présentes dans votre file d'attente à un moment donné

La `MainRequestQueuePendingRequests` CloudWatch métrique enregistre le nombre de demandes en attente dans la file d'entrée avec une granularité de cinq minutes (voirMétriques Neptune CloudWatch).

Pour Gremlin, vous pouvez obtenir le nombre actuel de requêtes dans la file d'attente en utilisant la valeur `acceptedQueryCount` renvoyée par l'API de statut des requêtes Gremlin. Notez toutefois que la valeur `acceptedQueryCount` renvoyée par l'API de statut des requêtes SPARQL inclut toutes les requêtes acceptées depuis le démarrage du serveur, y compris les requêtes terminées.

Comment la mise en file d'attente des requêtes peut affecter les délais d'expiration

Comme indiqué précédemment, la latence de requête inclut le temps passé par une requête dans la file d'attente, ainsi que le temps d'exécution.

Étant donné que le délai d'expiration d'une requête est généralement mesuré à partir du moment où elle entre dans la file d'attente, une file d'attente lente peut faire expirer de nombreuses requêtes

dès qu'elles sont retirées de la file d'attente. Ceci n'est évidemment pas souhaitable. Il est donc bon d'éviter de mettre en file d'attente un grand nombre de requêtes, à moins qu'elles ne puissent être exécutées rapidement.

Cache du plan de requête dans Amazon Neptune

Lorsqu'une requête est soumise à Neptune, la chaîne de requête est analysée, optimisée et transformée en un plan de requête, qui est ensuite exécuté par le moteur. Les applications sont souvent soutenues par des modèles de requête courants instanciés avec des valeurs différentes. Le cache des plans de requêtes peut réduire la latence globale en mettant en cache les plans de requêtes et en évitant ainsi l'analyse et l'optimisation pour de tels modèles répétés.

Le cache du plan de requête peut être utilisé pour les OpenCypherrequêtes, qu'elles soient paramétrées ou non paramétrées. Il est activé pour READ, HTTP et Bolt. Il n'est pas pris en charge pour les requêtes de mutation OC. Il n'est pas pris en charge pour les requêtes Gremlin ou SPARQL.

Comment forcer l'activation ou la désactivation du cache du plan de requêtes

Le cache du plan de requêtes est activé par défaut pour les requêtes paramétrées à faible latence. Un plan pour une requête paramétrée est mis en cache uniquement lorsque la latence est inférieure au seuil de 100 ms. Ce comportement peut être remplacé par requête (paramétré ou non) par le Query Hint au niveau de la requête. `QUERY:PLANCACHE` Il doit être utilisé avec la `USING` clause. L'indice de requête accepte `enabled` ou `disabled` en tant que valeur.

AWS CLI

Forcer le plan à être mis en cache ou réutilisé :

```
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1"
```

Avec paramètres :

```
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "Using QUERY:PLANCACHE \"enabled\" RETURN \$arg" \
```

```
  --parameters '{"arg": 123}'
```

Forcer le plan à ne pas être mis en cache ni réutilisé :

```
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "Using QUERY:PLANCACHE \"disabled\" MATCH(n) RETURN n LIMIT 1"
```

Pour plus d'informations, voir execute-open-cypher-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

# Forcing plan to be cached or reused
response = client.execute_open_cypher_query(
    openCypherQuery='Using QUERY:PLANCACHE "enabled" MATCH(n) RETURN n LIMIT 1'
)

print(response['results'])
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

Forcer le plan à être mis en cache ou réutilisé :

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "query=Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

Forcer le plan à être mis en cache ou réutilisé :

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1"
```

Avec paramètres :

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" RETURN \$arg" \
  -d "parameters={\"arg\": 123}"
```

Forcer le plan à ne pas être mis en cache ni réutilisé :

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"disabled\" MATCH(n) RETURN n LIMIT 1"
```

Comment déterminer si un plan est mis en cache ou non

Pour HTTP READ, si la requête a été soumise et que le plan a été mis en cache, `explain` les détails relatifs au cache du plan de requête seront affichés.

AWS CLI

```
aws neptunedata execute-open-cypher-explain-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1"
 \
  --explain-mode details
```

Pour plus d'informations, consultez execute-open-cypher-explain-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_explain_query(
    openCypherQuery='Using QUERY:PLANCACHE "enabled" MATCH(n) RETURN n LIMIT 1',
    explainMode='details'
)

print(response['results'].read().decode('utf-8'))
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "query=Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1" \
  -d "explain=details"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1" \
  -d "explain=details"
```

Si le plan a été mis en cache, le `explain` résultat indique :

```
Query: <QUERY STRING>
Plan cached by request: <REQUEST ID OF FIRST TIME EXECUTION>
Plan cached at: <TIMESTAMP OF FIRST TIME EXECUTION>
Parameters: <PARAMETERS, IF QUERY IS PARAMETERIZED QUERY>
Plan cache hits: <NUMBER OF CACHE HITS FOR CACHED PLAN>
First query evaluation time: <LATENCY OF FIRST TIME EXECUTION>

The query has been executed based on a cached query plan. Detailed explain with
 operator runtime statistics can be obtained by running the query with plan cache
 disabled (using HTTP parameter planCache=disabled).
```

Lorsque vous utilisez Bolt, la fonction d'explication n'est pas prise en charge.

Expulsion

Un plan de requête est supprimé en fonction de la durée de vie du cache (TTL) ou lorsqu'un nombre maximum de plans de requêtes mis en cache est atteint. Lorsque le plan de requête est atteint, le TTL est actualisé. Les valeurs par défaut sont les suivantes :

- 1000 : nombre maximum de plans pouvant être mis en cache par instance.
- TTL - 300 000 millisecondes ou 5 minutes. L'accès au cache redémarre le TTL et le réinitialise à 5 minutes.

Conditions empêchant le plan d'être mis en cache

Le cache du plan de requête ne serait pas utilisé dans les conditions suivantes :

1. Lorsqu'une requête est soumise à l'aide de l'indice de requête`QUERY:PLANCACHE "disabled"`. Vous pouvez réexécuter la requête et la supprimer `QUERY:PLANCACHE "disabled"` pour activer le cache du plan de requête.
2. Si la requête soumise n'est pas une requête paramétrée et ne contient pas l'indice. `QUERY:PLANCACHE "enabled"`
3. Si le temps d'évaluation de la requête est supérieur au seuil de latence, la requête n'est pas mise en cache et est considérée comme une requête de longue durée qui ne bénéficierait pas du cache du plan de requêtes.
4. Si la requête contient un modèle qui ne renvoie aucun résultat.

- c'est-à-dire `MATCH (n:nonexistentLabel) return n` lorsqu'il n'y a aucun nœud avec l'étiquette spécifiée.
- c'est-à-dire `MATCH (n {name: $param}) return n parameters={"param": "abcde"}` lorsqu'il n'y a aucun nœud contenant`name=abcde`.

5. Si le paramètre de requête est un type composite, tel que a `list` ou `map` a.

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" RETURN \$arg" \
  -d "parameters={\"arg\": [1, 2, 3]}"

curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" RETURN \$arg" \
  -d "parameters={\"arg\": {\"a\": 1}}"
```

6. Si le paramètre de requête est une chaîne qui n'a pas fait partie d'une opération de chargement ou d'insertion de données. Par exemple, s'il `CREATE (n {name: "X"})` est exécuté pour insérer`"X"`, `RETURN "X"` il est mis en cache, alors qu'`RETURN "Y"`il ne le serait pas, car il n'`"Y"`a pas été inséré et n'existe pas dans la base de données.

Injection d'un ID personnalisé dans une requête Neptune Gremlin ou SPARQL

Par défaut, Neptune attribue une valeur `queryId` unique à chaque requête. Vous pouvez utiliser cet ID pour obtenir des informations sur une requête en cours d'exécution (voir API de statut des requêtes Gremlin ou API de statut des requêtes SPARQL) ou pour l'annuler (voir Annulation de requêtes Gremlin ou Annulation de requêtes SPARQL).

Neptune vous permet également de spécifier votre propre valeur `queryId` pour une requête Gremlin ou SPARQL, soit dans l'en-tête HTTP, soit pour une requête SPARQL à l'aide de l'indicateur de requête `queryId`. L'attribution de votre propre `queryID` permet de suivre facilement une requête afin d'obtenir son statut ou de l'annuler.

Injection d'une valeur **`QueryID`** personnalisée à l'aide de l'en-tête HTTP

Pour Gremlin et SPARQL, l'en-tête HTTP peut être utilisé pour injecter votre propre valeur `queryId` dans une requête.

Exemple Gremlin

```
curl -XPOST https://your-neptune-endpoint:port \
    -d "{\"gremlin\": \
        \"g.V().limit(1).count()\" , \
        \"queryId\":\"4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47\"  }"
```

Exemple SPARQL

```
curl https://your-neptune-endpoint:port/sparql \
    -d "query=SELECT * WHERE { ?s ?p ?o } " \
       --data-urlencode \
       "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Injection d'une valeur **`QueryID`** personnalisée à l'aide d'un indice de requête SPARQL

Voici un exemple d'utilisation de l'indicateur de requête `queryId` SPARQL pour injecter une valeur `queryId` personnalisée dans une requête SPARQL :

```
curl https://your-neptune-endpoint:port/sparql \
    -d "query=PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#> \
       SELECT * WHERE { hint:Query hint:queryId \"4d5c4fae-
aa30-41cf-9e1f-91e6b7dd6f47\" \
       {?s ?p ?o}}"
```

Utilisation de la valeur **`QueryID`** pour vérifier l'état de la requête

Exemple Gremlin

```
curl https://your-neptune-endpoint:port/gremlin/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Exemple SPARQL

```
curl https://your-neptune-endpoint:port/sparql/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Accès au graphe Neptune avec Gremlin

Amazon Neptune est compatible avec Apache TinkerPop et Gremlin. Cela signifie que vous pouvez vous connecter à une instance de base de données Neptune et utiliser le langage de traversée G705 pour interroger le graphe (voir The Graph dans la documentation d'Apache). TinkerPop Pour voir les différences dans l'implémentation Neptune de Gremlin, consultez Conformité avec les normes Gremlin.

Une traversée dans Gremlin est une série d'étapes chaînées. Elle commence à un sommet (ou arc). Elle parcourt le graphe en suivant les arêtes extérieures de chaque sommet, puis les arêtes extérieures de ces sommets. Chaque étape représente une opération de la traversée. Pour plus d'informations, consultez The Traversal dans la TinkerPop documentation.

Les différentes versions du moteur Neptune prennent en charge différentes versions de Gremlin. Consultez la page de version du moteur de la version de Neptune que vous utilisez pour déterminer quelle version de Gremlin est prise en charge ou consultez le tableau suivant qui répertorie les versions les plus anciennes et les plus récentes TinkerPop prises en charge par les différentes versions du moteur Neptune :

Version du moteur Neptune	TinkerPop Version minimale	TinkerPop Version maximale
`1.3.2.0 and newer`	`3.7.1`	`3.7.3`
`1.3.1.0`	`3.6.2`	`3.6.5`
`1.3.0.0`	`3.6.2`	`3.6.4`
`1.2.1.0 <= 1.2.1.2`	`3.6.2`	`3.6.2`
`1.1.1.0 <= 1.2.0.2`	`3.5.5`	`3.5.6`
`1.1.0.0 and older`	`(deprecated)`	`(deprecated)`

TinkerPop les clients sont généralement rétrocompatibles au sein d'une série (par exemple`3.6.x`, ou`3.7.x`) et bien qu'ils puissent souvent travailler au-delà de ces limites, le tableau ci-dessus recommande les combinaisons de versions à utiliser pour une expérience et une compatibilité optimales. Sauf avis contraire, il est généralement préférable de respecter ces directives et de mettre

à niveau les applications clientes pour qu'elles correspondent à la version TinkerPop que vous utilisez.

Lors de la mise à niveau des TinkerPop versions, il est toujours important TinkerPopde consulter la documentation de mise à niveau qui vous aidera à identifier les nouvelles fonctionnalités dont vous pouvez tirer parti, mais également les problèmes dont vous devrez peut-être être conscient à l'approche de votre mise à niveau. Vous devez généralement vous attendre à ce que les requêtes et fonctionnalités existantes fonctionnent après la mise à niveau, sauf si un problème particulier est indiqué comme un problème à prendre en compte. Enfin, il est important de noter que si vous mettez à niveau une version pour intégrer une nouvelle fonctionnalité, vous ne pourrez peut-être pas l'utiliser si elle provient d'une version ultérieure à celle prise en charge par Neptune.

Il existe des variantes du langage Gremlin et une prise en charge de l'accès Gremlin dans les différents langages de programmation. Pour plus d'informations, voir À propos des variantes linguistiques de Gkremlin dans la TinkerPop documentation.

Cette documentation décrit comment accéder à Neptune avec les variantes et langages de programmation suivants :

- Configuration de la console Gremlin pour se connecter à une instance de base de données Neptune
- Utilisation du point de terminaison HTTP REST pour se connecter à une instance de base de données Neptune
- Java-based Clients Gkremlin à utiliser avec Amazon Neptune
- Utilisation de Python pour se connecter à une instance de base de données Neptune
- Utilisation de .NET pour se connecter à une instance de base de données Neptune
- Utilisation Node.js pour se connecter à une instance de base de données Neptune
- Utilisation de Go pour se connecter à une instance de base de données Neptune

Comme indiqué dans la sectionChiffrer les connexions à votre base de données Amazon Neptune avec SSL/HTTPS, vous devez utiliser le protocole Transport Security/Secure Layer Sockets Layer (TLS/SSL) lorsque vous vous connectez à Neptune dans toutes les AWS régions.

Avant de commencer, les prérequis suivants doivent être remplis :

- Vous devez disposer d'une instance de base de données Neptune. Pour plus d'informations sur la création d'une instance de base de données Neptune, consultez Création d'un cluster Amazon Neptune.
- Vous devez disposer d'une instance Amazon EC2 dans le même cloud privé virtuel (VPC) que l'instance de base de données Neptune.

Pour plus d'informations sur le chargement des données dans Neptune, y compris les prérequis, les formats de chargement et les paramètres de chargement, consultez Chargement de données dans Amazon Neptune.

Rubriques

- Configuration de la console Gremlin pour se connecter à une instance de base de données Neptune
- Utilisation du point de terminaison HTTP REST pour se connecter à une instance de base de données Neptune
- Java-based Clients Gkremlin à utiliser avec Amazon Neptune
- Utilisation de Python pour se connecter à une instance de base de données Neptune
- Utilisation de .NET pour se connecter à une instance de base de données Neptune
- Utilisation Node.js pour se connecter à une instance de base de données Neptune
- Utilisation de Go pour se connecter à une instance de base de données Neptune
- Utilisation de AWS SDK pour exécuter des requêtes G705
- Indicateurs de requête Gremlin
- API de statut des requêtes Gremlin
- Annulation de requêtes Gremlin
- Prise en charge des sessions basées sur des scripts Gremlin
- Transactions Gremlin dans Neptune
- Diffusion en continu des résultats de requêtes avec Gremlin
- Utilisation de l'API Gremlin avec Amazon Neptune
- Mise en cache des résultats de requête dans Amazon Neptune Gremlin
- Réaliser des upserts efficaces avec les étapes Gremlin MergeV () et MergeE ()
- Réaliser des upserts Gmail efficaces avec fold () /coalesce () /unfold ()

- Analyse de l'exécution des requêtes Neptune à l'aide de Gremlin (explication)
- Utilisation de Gremlin avec le moteur de requêtes Neptune DFE

Configuration de la console Gremlin pour se connecter à une instance de base de données Neptune

La console G705 vous permet d'expérimenter avec TinkerPop des graphiques et des requêtes dans un environnement REPL (read-eval-print loop).

Installation de la console Gremlin et mode de connexion habituel

Vous pouvez utiliser la console Gremlin pour vous connecter à une base de données distante orientée graphe. La section suivante vous guide dans l'installation et la configuration de la console Gremlin pour une connexion à distance à une instance de base de données Neptune. Vous devez suivre ces instructions à partir d'une instance Amazon EC2 dans le même cloud privé virtuel (VPC) (VPC) que l'instance de base de données Neptune.

Pour obtenir de l'aide sur la connexion à Neptune avec SSL/TLS (obligatoire), consultez. SSL/TLS configuration

> **Note**
>
> Si l'authentification IAM est activée sur votre cluster de bases de données Neptune, suivez les instructions décrites dans Connexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec la console Gremlin pour installer la console Gremlin plutôt que celles indiquées ici.

Pour installer la console Gremlin et vous connecter à Neptune

1. Les fichiers binaires de la console Gremlin nécessitent Java 8 ou Java 11. Ces instructions supposent l'utilisation de Java 11. Vous pouvez installer Java 11 sur votre instance EC2 comme suit :

   - Si vous utilisez Amazon Linux 2 (AL2) :

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

- Si vous utilisez Amazon Linux 2023 (AL2023) :

```
sudo yum install java-11-amazon-corretto-devel
```

- Pour les autres distributions, utilisez l'instruction qui convient le mieux parmi les suivantes :

```
sudo yum install java-11-openjdk-devel
```

ou :

```
sudo apt-get install openjdk-11-jdk
```

2. Entrez la commande suivante pour définir Java 8 en tant qu'environnement d'exécution par défaut sur votre instance EC2.

```
sudo /usr/sbin/alternatives --config java
```

Lorsque vous y êtes invité, saisissez le nombre correspondant à Java 11.

3. Téléchargez la version appropriée de la console Gremlin depuis le site Web d'Apache. Vous pouvez vérifier la version Accès au graphe Neptune avec Gremlin de Gkremlin prise en charge par votre version de Neptune. Par exemple, si vous avez besoin de la version 3.7.2, vous pouvez télécharger la console G705 depuis le site Web d'Apache Tinkerpop sur votre instance EC2 comme suit :

```
wget https://archive.apache.org/dist/tinkerpop/3.7.2/apache-tinkerpop-gremlin-
console-3.7.2-bin.zip
```

4. Décompressez le fichier zip Gremlin Console.

```
unzip apache-tinkerpop-gremlin-console-3.7.2-bin.zip
```

5. Modifiez les répertoires du répertoire décompressé.

```
cd apache-tinkerpop-gremlin-console-3.7.2
```

6. Dans le sous-répertoire `conf` du répertoire extrait, créez un fichier nommé `neptune-remote.yaml` avec le texte suivant. *`your-neptune-endpoint`*Remplacez-le par le nom d'hôte ou l'adresse IP de votre instance de base de données Neptune. Les crochets (`[ ]`) sont obligatoires.

> **Note**
>
> Consultez la section Connexion aux points de terminaison Amazon Neptune pour découvrir comment trouver le nom d'hôte de votre instance de base de données Neptune.

```
hosts: [your-neptune-endpoint]
port: 8182
connectionPool: { enableSsl: true }
serializer: { className:
 org.apache.tinkerpop.gremlin.util.ser.GraphBinaryMessageSerializerV1,
              config: { serializeResultToString: true }}
```

> **Note**
>
> Les sérialiseurs ont été déplacés du `gremlin-driver` module vers le nouveau `gremlin-util` module dans la version 3.7.0. Le package est passé de org.apache.tinkerpop.gremlin.driver.ser à org.apache.tinkerpop.gremlin.util.ser.

7. Dans un terminal, accédez au répertoire Gremlin Console (`apache-tinkerpop-gremlin-console-3.7.2`), puis entrez la commande suivante pour exécuter la console Gremlin.

```
bin/gremlin.sh
```

Vous devriez voir la sortie suivante :

```
         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin>
```

Vous êtes maintenant à l'invite `gremlin>`. Vous entrez les étapes restantes à cette invite.

8. À l'invite de commande `gremlin>`, saisissez le texte suivant pour vous connecter à l'instance de base de données Neptune.

```
:remote connect tinkerpop.server conf/neptune-remote.yaml
```

9. À l'invite `gremlin>`, entrez ce qui suit pour passer en mode distant. Toutes les requêtes Gremlin sont alors envoyées à la connexion distante.

```
:remote console
```

10. Saisissez la commande suivante pour envoyer une requête au graphe Gremlin.

```
g.V().limit(1)
```

11. Lorsque vous avez terminé, saisissez la commande suivante pour quitter la console Gremlin.

```
:exit
```

> **Note**
>
> Utilisez un point-virgule (`;`) ou un caractère de saut de ligne (`\n`) pour séparer chaque instruction.
> Chaque traversée précédant la traversée finale doit se terminer par l'exécution de `next()`. Seules les données de la traversée finale sont renvoyées.

Pour plus d'informations sur l'implémentation Neptune de Gremlin, consultez the section called "Conformité avec les normes Gremlin".

Autre mode de connexion à la console Gremlin

Inconvénients de l'approche de connexion habituelle

La méthode la plus courante pour se connecter à la console Gremlin est expliquée ci-dessus et utilise des commandes comme celle-ci à l'invite `gremlin>` :

```
gremlin> :remote connect tinkerpop.server conf/(file name).yaml
gremlin> :remote console
```

Elle est opérationnelle et vous permet d'envoyer des requêtes à Neptune. Cependant, elle exclut le moteur de script Groovy, de sorte que Neptune traite toutes les requêtes comme si elles étaient entièrement écrites dans Gremlin. Par conséquent, les types de requête suivants échouent :

```
gremlin> 1 + 1
gremlin> x = g.V().count()
```

Le meilleur moyen d'utiliser une variable lorsque vous êtes connecté de cette façon consiste à utiliser la variable `result` gérée par la console et d'envoyer la requête en utilisant `:>`, comme ceci :

```
gremlin> :remote console
==>All scripts will now be evaluated locally - type ':remote console' to return
 to remote mode for Gremlin Server - [krl-1-cluster.cluster-ro-cm9t6tfwbtsr.us-
east-1.neptune.amazonaws.com/172.31.19.217:8182]
gremlin> :> g.V().count()
==>4249

gremlin> println(result)
[result{object=4249 class=java.lang.Long}]

gremlin> println(result['object'])
[4249]
```

Autre mode de connexion

Vous pouvez également vous connecter à la console Gremlin d'une autre manière, que vous trouverez peut-être plus agréable, comme ceci :

```
gremlin> g = traversal().withRemote('conf/neptune.properties')
```

`neptune.properties` prend ici la forme suivante :

```
gremlin.remote.remoteConnectionClass=org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteCon
gremlin.remote.driver.clusterFile=conf/my-cluster.yaml
gremlin.remote.driver.sourceName=g
```

Le fichier `my-cluster.yaml` doit ressembler à ceci :

```
hosts: [my-cluster-abcdefghijk.us-east-1.neptune.amazonaws.com]
```

```
port: 8182
serializer: { className:
 org.apache.tinkerpop.gremlin.util.ser.GraphBinaryMessageSerializerV1,
              config: { serializeResultToString: false } }
connectionPool: { enableSsl: true }
```

> **Note**
>
> Les sérialiseurs ont été déplacés du `gremlin-driver` module vers le nouveau `gremlin-util` module dans la version 3.7.0. Le package est passé de org.apache.tinkerpop.gremlin.driver.ser à org.apache.tinkerpop.gremlin.util.ser.

En configurant ainsi la connexion à la console Gremlin, vous pouvez effectuer avec succès les types de requêtes suivants :

```
gremlin> 1+1
==>2

gremlin> x=g.V().count().next()
==>4249

gremlin> println("The answer was ${x}")
The answer was 4249
```

Pour éviter d'afficher le résultat, procédez comme suit :

```
gremlin> x=g.V().count().next();[]
gremlin> println(x)
4249
```

Toutes les méthodes habituelles d'interrogation (sans l'étape terminale) continuent de fonctionner. Par exemple :

```
gremlin> g.V().count()
==>4249
```

Vous pouvez même utiliser l'étape `g.io().read()` pour charger un fichier avec ce type de connexion.

Authentification IAM

Neptune prend en charge l'authentification IAM pour contrôler l'accès à votre cluster de base de données. Si l'authentification IAM est activée, vous devez utiliser la signature Signature version 4 pour authentifier vos demandes. Pour obtenir des instructions détaillées et des exemples de code pour la connexion depuis la console Gremlin, consultezConnexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec la console Gremlin.

Utilisation du point de terminaison HTTP REST pour se connecter à une instance de base de données Neptune

Amazon Neptune fournit un point de terminaison HTTP pour les requêtes Gremlin. L'interface REST est compatible avec n'importe quelle version Gremlin de votre cluster de bases de données (consultez la page de mise à jour correspondant à la version de moteur Neptune que vous exécutez pour déterminer la version Gremlin prise en charge).

> **Note**
>
> Comme indiqué dans Chiffrer les connexions à votre base de données Amazon Neptune avec SSL/HTTPS, vous devez désormais vous connecter via HTTPS au lieu de HTTP dans Neptune. En outre, Neptune ne prend actuellement pas en charge HTTP/2 les requêtes d'API REST. Les clients doivent l'utiliser HTTP/1.1 lorsqu'ils se connectent aux points de terminaison.

Les instructions suivantes vous guident à travers la connexion au point de terminaison Gremlin à l'aide de la commande `curl` et de HTTPS. Vous devez suivre ces instructions à partir d'une instance Amazon EC2 dans le même cloud privé virtuel (VPC) (VPC) que l'instance de base de données Neptune.

Le point de terminaison HTTPS pour les requêtes Gremlin dans une instance de base de données Neptune est `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin`.

> **Note**
>
> Pour découvrir comment trouver le nom d'hôte de l'instance de base de données Neptune, consultez la section Connexion aux points de terminaison Amazon Neptune.

Pour se connecter à Neptune à l'aide du point de terminaison HTTP REST

Les exemples suivants montrent comment envoyer une requête Gremlin au point de terminaison REST. Vous pouvez utiliser le AWS SDK AWS CLI, le oucurl.

AWS CLI

```
aws neptunedata execute-gremlin-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --gremlin-query "g.V().limit(1)"
```

Pour plus d'informations, consultez execute-gremlin-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
import json
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_gremlin_query(
    gremlinQuery='g.V().limit(1)',
    serializer='application/vnd.gremlin-v3.0+json;types=false'
)

print(json.dumps(response['result'], indent=2))
```

Pour des exemples de AWS SDK dans d'autres langages tels que Java, .NET, etc., consultezAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/gremlin \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
```

```
  -d '{"gremlin":"g.V().limit(1)"}'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultezUtiliser awscurl avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

curl

L'exemple suivant utilise curl pour soumettre une requête Gremlin via HTTP POST. La requête est soumise au format JSON dans le corps de la publication en tant que propriété `gremlin`.

```
curl -X POST -d '{"gremlin":"g.V().limit(1)"}' https://your-neptune-endpoint:port/
gremlin
```

Bien que les demandes HTTP POST soient recommandées pour envoyer des requêtes Gremlin, il est également possible d'utiliser des demandes HTTP GET :

```
curl -G "https://your-neptune-endpoint:port?gremlin=g.V().count()"
```

Ces exemples renvoient le premier sommet du graphe à l'aide de la `g.V().limit(1)` traversée. Pour interroger autre chose, remplacez cette traversée par une autre traversée Gremlin.

> **Important**
>
> Par défaut, le point de terminaison REST renvoie tous les résultats dans un seul ensemble de résultats JSON. Si cet ensemble de résultats est trop volumineux, une exception `OutOfMemoryError` peut être générée dans l'instance de base de données Neptune. Pour éviter cette exception, activez les réponses segmentées (résultats renvoyés sous la forme d'une série de réponses distinctes). Consultez Utilisation d'en-têtes de suivi HTTP facultatifs pour activer les réponses Gremlin en plusieurs parties.

> **Note**
>
> Neptune ne prend pas en charge la propriété `bindings`.

Utilisation d'en-têtes de suivi HTTP facultatifs pour activer les réponses Gremlin en plusieurs parties

Par défaut, la réponse HTTP aux requêtes Gremlin est renvoyée dans un seul ensemble de résultats JSON. Un ensemble de résultats très volumineux peut générér une exception `OutOfMemoryError` dans l'instance de base de données.

Vous pouvez toutefois activer les réponses segmentées (réponses renvoyées en plusieurs parties distinctes). Pour ce faire, incluez un en-tête « trailer » (`te: trailers`) d'encodage de transfert (TE) dans la demande. Consultez la page MDN (sur les en-têtes de demande TE) pour plus d'informations sur ces en-têtes.

Lorsqu'une réponse est renvoyée en plusieurs parties, il peut être difficile de diagnostiquer un problème qui survient après la réception de la première partie, car la première partie arrive avec le code de statut HTTP `200` (OK). En cas d'échec ultérieur, le corps du message contient généralement une réponse corrompue, à la fin de laquelle Neptune ajoute un message d'erreur.

Pour faciliter la détection et le diagnostic de ce type d'échec, Neptune inclut également deux nouveaux champs d'en-tête dans les en-têtes suivants de chaque segment de réponse :

- `X-Neptune-Status` : contient le code de réponse suivi d'un nom court. Par exemple, en cas de réussite, l'en-tête final serait : `X-Neptune-Status: 200 OK`. En cas d'échec, le code de réponse est l'un des codes d'erreur du moteur Neptune, tel que `X-Neptune-Status: 500 TimeLimitExceededException`.
- `X-Neptune-Detail` : est vide pour les demandes qui ont abouti. En cas d'erreur, il contient le message d'erreur JSON. Étant donné que seuls les caractères ASCII sont autorisés dans les valeurs d'en-tête HTTP, la chaîne JSON est encodée en URL.

> **Note**
>
> Neptune ne prend actuellement pas en charge la compression `gzip` des réponses segmentées. Si le client demande à la fois un encodage et une compression segmentés, Neptune ignore la compression.

Java-based Clients Gkremlin à utiliser avec Amazon Neptune

Vous pouvez utiliser l'un des deux clients Java-based G705 open source avec Amazon Neptune : le client TinkerPop Apache Java G705 ou le client G705 pour Amazon Neptune.

Client Apache TinkerPop Java Gremlin

Le pilote gremlin Apache TinkerPop Java est le client Gremlin standard et officiel qui fonctionne avec n'importe quelle base de données de graphes. TinkerPop-enabled Utilisez ce client lorsque vous avez besoin d'une compatibilité maximale avec l'ensemble de l'espace de TinkerPop développement, lorsque vous travaillez avec plusieurs systèmes de base de données de graphes ou lorsque vous n'avez pas besoin des fonctionnalités avancées de gestion des clusters et d'équilibrage de charge spécifiques à Neptune. Ce client convient également aux applications simples qui se connectent à une seule instance Neptune ou lorsque vous préférez gérer l'équilibrage de charge au niveau de l'infrastructure plutôt qu'au sein du client.

> **Important**
>
> Le choix de la bonne version du pilote TinkerPop Apache G705 est essentiel pour garantir la compatibilité avec la version de votre moteur Neptune. L'utilisation d'une version incompatible peut entraîner des échecs de connexion ou un comportement inattendu. Pour obtenir des informations détaillées sur la compatibilité des versions, consultezAccès au graphe Neptune avec Gremlin.

> **Note**
>
> Le tableau qui vous aide à déterminer la TinkerPop version correcte d'Apache à utiliser avec Neptune a été déplacé vers. the section called “Gremlin” Ce tableau se trouvait auparavant

sur cette page depuis de nombreuses années et est désormais plus centralisé à titre de référence pour tous les langages de programmation pris TinkerPop en charge.

Client Java Gremlin pour Amazon Neptune

Le client G705 pour Amazon Neptune est un client Java-based G705 open source qui remplace directement le client Java standard. TinkerPop

Le client Gremlin Neptune est optimisé pour les clusters Neptune. Il vous permet de gérer la distribution du trafic entre plusieurs instances d'un cluster et s'adapte aux modifications de la topologie du cluster lorsque vous ajoutez ou supprimez un réplica. Vous pouvez même configurer le client pour distribuer les demandes sur un sous-ensemble d'instances du cluster, en fonction du rôle, du type d'instance, de la zone de disponibilité (AZ) ou des balises associées aux instances.

La dernière version du client Java Gremlin Neptune est disponible sur Maven Central.

Pour plus d'informations sur le client Java Gremlin Neptune, consultez ce billet de blog. Pour des exemples de code et des démos, consultez le GitHub projet du client.

Lorsque vous choisissez la version du client Neptune Gkremlin, vous devez prendre en compte la TinkerPop version sous-jacente par rapport à la version de votre moteur Neptune. Reportez-vous au tableau de compatibilité sur the section called "Gremlin" pour déterminer la TinkerPop version appropriée pour votre moteur Neptune, puis utilisez le tableau suivant pour sélectionner la version du client Neptune Gremlin appropriée :

Compatibilité des versions du client Neptune G705

Version du client Neptune Garmlin	TinkerPop version
3.x	3.7.x (AWS SDK pour Java 2. x/1. x)
2.1.x	3.7.x (AWS SDK pour Java 1.x)
2.0.x	3,6. x
1.12	3.5.x

Utilisation du client Java pour se connecter à une instance de base de données Neptune

La section suivante explique comment exécuter un exemple Java complet qui se connecte à une instance de base de données Neptune et effectue une traversée de G705 à l'aide du client Apache G705. TinkerPop

Vous devez suivre ces instructions à partir d'une instance Amazon EC2 située dans le même cloud privé virtuel (VPC) (VPC) que l'instance de base de données Neptune.

Pour se connecter à Neptune à l'aide de Java

1. Installez Apache Maven sur votre instance EC2. Si vous utilisez Amazon Linux 2023 (de préférence), utilisez :

   ```
   sudo dnf update -y
   sudo dnf install maven -y
   ```

   Si vous utilisez Amazon Linux 2, téléchargez le dernier fichier binaire à l'adresse suivante https://maven.apache.org/download.cgi:

   ```
   sudo yum remove maven -y
   wget https://dlcdn.apache.org/maven/maven-3/ <version>/binaries/apache-maven-
   <version>-bin.tar.gz
   sudo tar -xzf apache-maven-<version>-bin.tar.gz -C /opt/
   sudo ln -sf /opt/apache-maven-<version> /opt/maven
   echo 'export MAVEN_HOME=/opt/maven' >> ~/.bashrc
   echo 'export PATH=$MAVEN_HOME/bin:$PATH' >> ~/.bashrc
   source ~/.bashrc
   ```

2. Installez Java. Les bibliothèques Gremlin nécessitent Java 8 ou 11. Vous pouvez installer Java 11 comme suit :

   - Si vous utilisez Amazon Linux 2 (AL2) :

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

   - Si vous utilisez Amazon Linux 2023 (AL2023) :

     ```
     sudo yum install java-11-amazon-corretto-devel
     ```

- Pour les autres distributions, utilisez l'instruction qui convient le mieux parmi les suivantes :

```
sudo yum install java-11-openjdk-devel
```

ou :

```
sudo apt-get install openjdk-11-jdk
```

3. Définissez Java 11 comme environnement d'exécution par défaut sur votre instance EC2 : entrez ce qui suit pour définir Java 8 comme environnement d'exécution par défaut sur l'instance EC2 :

```
sudo /usr/sbin/alternatives --config java
```

Lorsque vous y êtes invité, saisissez le nombre correspondant à Java 11.

4. Créez un répertoire appelé **`gremlinjava`** :

```
mkdir gremlinjava
cd gremlinjava
```

5. Dans le répertoire `gremlinjava`, créez un fichier `pom.xml`, puis ouvrez-le dans un éditeur de texte:

```
nano pom.xml
```

6. Copiez ce qui suit dans le fichier `pom.xml` et enregistrez-le:

```
<project xmlns="https://maven.apache.org/POM/4.0.0"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://
maven.apache.org/maven-v4_0_0.xsd">
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.amazonaws</groupId>
  <artifactId>GremlinExample</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>GremlinExample</name>
  <url>https://maven.apache.org</url>
```

```
  <dependencies>
    <dependency>
      <groupId>org.apache.tinkerpop</groupId>
      <artifactId>gremlin-driver</artifactId>
      <version>3.7.2</version>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-jdk14</artifactId>
      <version>1.7.25</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.5.1</version>
        <configuration>
          <source>11</source>
          <target>11</target>
        </configuration>
      </plugin>
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>exec-maven-plugin</artifactId>
          <version>1.3</version>
          <configuration>
            <executable>java</executable>
            <arguments>
              <argument>-classpath</argument>
              <classpath/>
              <argument>com.amazonaws.App</argument>
            </arguments>
            <mainClass>com.amazonaws.App</mainClass>
            <complianceLevel>1.11</complianceLevel>
            <killAfter>-1</killAfter>
          </configuration>
        </plugin>
    </plugins>
  </build>
</project>
```

> **Note**
>
> Si vous modifiez un projet Maven existant, la dépendance obligatoire est mise en évidence dans le code précédent.

7. Créez des sous-répertoires pour l'exemple de code source (`src/main/java/com/amazonaws/`) en saisissant ce qui suit sur la ligne de commande :

```
mkdir -p src/main/java/com/amazonaws/
```

8. Dans le répertoire `src/main/java/com/amazonaws/`, créez un fichier `App.java`, puis ouvrez-le dans un éditeur de texte.

```
nano src/main/java/com/amazonaws/App.java
```

9. Copiez ce qui suit dans le fichier `App.java`. Remplacez *`your-neptune-endpoint`* par l'adresse de votre instance de base de données Neptune. N'incluez pas le préfixe `https://` dans la méthode `addContactPoint`.

> **Note**
>
> Pour découvrir comment trouver le nom d'hôte de l'instance de base de données Neptune, consultez la section Connexion aux points de terminaison Amazon Neptune.

```
package com.amazonaws;
import org.apache.tinkerpop.gremlin.driver.Cluster;
import
 org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversalSource;
import org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal;
import static
 org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource.traversal;
import org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteConnection;
import org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.__;
import org.apache.tinkerpop.gremlin.structure.T;

public class App
{
  public static void main( String[] args )
```

```
  {
    Cluster.Builder builder = Cluster.build();
    builder.addContactPoint("your-neptune-endpoint");
    builder.port(8182);
    builder.enableSsl(true);

    Cluster cluster = builder.create();

    GraphTraversalSource g =
 traversal().withRemote(DriverRemoteConnection.using(cluster));

    // Add a vertex.
    // Note that a Gremlin terminal step, e.g. iterate(), is required to make a
 request to the remote server.
    // The full list of Gremlin terminal steps is at https://tinkerpop.apache.org/
docs/current/reference/#terminal-steps
    g.addV("Person").property("Name", "Justin").iterate();

    // Add a vertex with a user-supplied ID.
    g.addV("Custom Label").property(T.id, "CustomId1").property("name", "Custom id
 vertex 1").iterate();
    g.addV("Custom Label").property(T.id, "CustomId2").property("name", "Custom id
 vertex 2").iterate();

    g.addE("Edge Label").from(__.V("CustomId1")).to(__.V("CustomId2")).iterate();

    // This gets the vertices, only.
    GraphTraversal t = g.V().limit(3).elementMap();

    t.forEachRemaining(
      e ->  System.out.println(t.toList())
    );

    cluster.close();
  }
}
```

Pour obtenir de l'aide sur la connexion à Neptune avec SSL/TLS (obligatoire), consultez. SSL/TLS configuration

10. Compilez et exécutez l'exemple à l'aide de la commande Maven suivante :

```
mvn compile exec:exec
```

L'exemple précédent renvoie une carte de la clé et les valeurs de chaque propriété pour les deux premiers vertex du graphe à l'aide de la traversée `g.V().limit(3).elementMap()`. Pour interroger quelque chose d'autre, remplacez la traversée par une autre traversée Gremlin avec l'une des méthodes de fin appropriées.

> **Note**
>
> La partie finale de la requête Gremlin, `.toList()`, est obligatoire pour soumettre la traversée au serveur à des fins d'évaluation. Si vous n'incluez pas cette méthode ou une autre méthode équivalente, la requête ne sera pas soumise à l'instance de base de données Neptune.
>
> Vous devez également ajouter une terminaison appropriée lorsque vous ajoutez un sommet ou une arête, comme lorsque vous utilisez l'étape `addV( )`.

Voici des exemples de méthodes qui soumettent la requête à l'instance de base de données Neptune :

- `toList()`
- `toSet()`
- `next()`
- `iterate()`

`iterate()`À utiliser lorsque vous n'avez pas besoin des résultats de vos requêtes (par exemple des mutations) car cela permet de réduire les coûts de sérialisation.

Ces étapes du terminal se comportent différemment selon que vous soumettez des requêtes sous forme de chaînes (mode script) ou d'objets de traversée de bytecode (mode bytecode). Les différences sont particulièrement importantes pour le traitement des transactions. Pour en savoir plus, consultez Testez le code Gremlin dans le contexte dans lequel vous allez le déployer.

SSL/TLS configuration pour le client Java Gkremlin

Neptune doit SSL/TLS être activé par défaut. Généralement, si le pilote Java est configuré avec `enableSsl(true)`, il peut se connecter à Neptune sans avoir à configurer `trustStore()` ou `keyStore()` avec une copie locale d'un certificat.

Toutefois, si l'instance à laquelle vous vous connectez ne dispose pas d'une connexion Internet permettant de vérifier un certificat public ou si le certificat que vous utilisez n'est pas public, vous pouvez suivre les étapes ci-dessous pour configurer une copie du certificat local :

Configuration d'une copie de certificat locale pour activer SSL/TLS

1. Téléchargez et installez keytool depuis Oracle. Cela facilitera grandement la configuration du magasin de clés local.
2. Téléchargez le certificat CA `SFSRootCAG2.pem` (le kit SDK Java Gremlin a besoin d'un certificat pour vérifier le certificat à distance).

   ```
   wget https://www.amazontrust.com/repository/SFSRootCAG2.pem
   ```

3. Créez un magasin de clés au format JKS ou PKCS12. Cet exemple utilise JKS. Répondez aux questions qui suivent à l'invite. Le mot de passe que vous créez ici sera nécessaire ultérieurement :

   ```
   keytool -genkey -alias (host name) -keyalg RSA -keystore server.jks
   ```

4. Importez le fichier `SFSRootCAG2.pem` que vous avez téléchargé dans le magasin de clés que vous venez de créer :

   ```
   keytool -import -keystore server.jks -file .pem
   ```

5. Configurez l'objet `Cluster` par programmation :

   ```
   Cluster cluster = Cluster.build("(your neptune endpoint)")
                            .port(8182)
                            .enableSSL(true)
                            .keyStore('server.jks')
                            .keyStorePassword("(the password from step 2)")
                            .create();
   ```

   Vous pouvez faire la même chose dans un fichier de configuration si vous le souhaitez, comme vous pourriez le faire avec la console Gremlin :

   ```
   hosts: [(your neptune endpoint)]
   port: 8182
   ```

```
connectionPool: { enableSsl: true, keyStore: server.jks, keyStorePassword: (the
 password from step 2) }
serializer: { className:
 org.apache.tinkerpop.gremlin.util.ser.GraphBinaryMessageSerializerV1, config:
 { serializeResultToString: true }}
```

Authentification IAM

Neptune prend en charge l'authentification IAM pour contrôler l'accès à votre cluster de base de données. Si l'authentification IAM est activée, vous devez utiliser la signature Signature version 4 pour authentifier vos demandes. Pour obtenir des instructions détaillées et des exemples de code pour la connexion à partir d'un client Java, consultezConnexion aux bases de données Amazon Neptune à l'aide d'IAM avec Gremlin Java.

Exemple Java de connexion à une instance de base de données Neptune avec une logique de reconnexion

L'exemple Java suivant montre comment se connecter au client Gremlin à l'aide d'une logique de reconnexion afin d'assurer la reprise après une déconnexion inattendue.

Pour obtenir des conseils détaillés sur le développement d'une stratégie de réessai pratique, y compris un recul exponentiel avec instabilité, voir. the section called “Exceptions et nouvelles tentatives”

Il présente les dépendances suivantes :

```
<dependency>
    <groupId>org.apache.tinkerpop</groupId>
    <artifactId>gremlin-driver</artifactId>
    <version>${gremlin.version}</version>
</dependency>

<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>amazon-neptune-sigv4-signer</artifactId>
    <version>${sig4.signer.version}</version>
</dependency>

<dependency>
    <groupId>com.evanlennick</groupId>
    <artifactId>retry4j</artifactId>
```

```
    <version>0.15.0</version>
</dependency>
```

Voici l'exemple de code :

> **Important**
>
> Le `CallExecutor` de Retry4j n'est peut-être pas thread-safe. Envisagez de demander à chaque thread d'utiliser sa propre `CallExecutor` instance ou d'utiliser une bibliothèque de nouvelles tentatives différente.

> **Note**
>
> L'exemple suivant a été mis à jour pour inclure l'utilisation de RequestInterceptor (). Cela a été ajouté dans la version TinkerPop 3.6.6. Avant la TinkerPop version 3.6.6, l'exemple de code utilisait HandshakeInterceptor (), qui était obsolète avec cette version.

```
public static void main(String args[]) {
  boolean useIam = true;

  // Create Gremlin cluster and traversal source
  Cluster.Builder builder = Cluster.build()
         .addContactPoint(System.getenv("neptuneEndpoint"))
         .port(Integer.parseInt(System.getenv("neptunePort")))
         .enableSsl(true)
         .minConnectionPoolSize(1)
         .maxConnectionPoolSize(1)
         .serializer(Serializers.GRAPHBINARY_V1D0)
         .reconnectInterval(2000);

  if (useIam) {
      builder.requestInterceptor( r -> {
         try {
            NeptuneNettyHttpSigV4Signer sigV4Signer =
                        new NeptuneNettyHttpSigV4Signer("(your region)", new
 DefaultAWSCredentialsProviderChain());
            sigV4Signer.signRequest(r);
         } catch (NeptuneSigV4SignerException e) {
```

```
            throw new RuntimeException("Exception occurred while signing the request",
e);
        }
        return r;
    });
  }

 Cluster cluster = builder.create();

 GraphTraversalSource g = AnonymousTraversalSource
     .traversal()
     .withRemote(DriverRemoteConnection.using(cluster));

 // Configure retries
 // NOTE: This example uses a fixed backoff for simplicity.
 // In a production application consider use of exponential backoff with jitter.
 RetryConfig retryConfig = new RetryConfigBuilder()
     .retryOnCustomExceptionLogic(getRetryLogic())
     .withDelayBetweenTries(1000, ChronoUnit.MILLIS)
     .withMaxNumberOfTries(5)
     .withFixedBackoff()
     .build();

 @SuppressWarnings("unchecked")
 CallExecutor<Object> retryExecutor = new CallExecutorBuilder<Object>()
     .config(retryConfig)
     .build();

 // Do lots of queries
 for (int i = 0; i < 100; i++){
   String id = String.valueOf(i);

   @SuppressWarnings("unchecked")
   Callable<Object> query = () -> g.mergeV(Map.of(T.id, id))
       .option(Merge.onCreate, Map.of(T.label, "Person"))
       .id().next();

   // Retry query
   // If there are connection failures, the Java Gremlin client will automatically
   // attempt to reconnect in the background, so all we have to do is wait and retry.
   Status<Object> status = retryExecutor.execute(query);

   System.out.println(status.getResult().toString());
 }
```

```
  cluster.close();
}

private static Function<Exception, Boolean> getRetryLogic() {

  return e -> {

    Class<? extends Exception> exceptionClass = e.getClass();

    StringWriter stringWriter = new StringWriter();
    e.printStackTrace(new PrintWriter(stringWriter));
    String message = stringWriter.toString();


    if (RemoteConnectionException.class.isAssignableFrom(exceptionClass)){
      System.out.println("Retrying because RemoteConnectionException");
      return true;
    }

    // Check for connection issues
    if (message.contains("Timed out while waiting for an available host") ||
        message.contains("Timed-out") && message.contains("waiting for connection on
 Host") ||
        message.contains("Connection to server is no longer active") ||
        message.contains("Connection reset by peer") ||
        message.contains("SSLEngine closed already") ||
        message.contains("Pool is shutdown") ||
        message.contains("ExtendedClosedChannelException") ||
        message.contains("Broken pipe") ||
        message.contains(System.getenv("neptuneEndpoint")))
    {
      System.out.println("Retrying because connection issue");
      return true;
    };

    // Concurrent writes can sometimes trigger a ConcurrentModificationException.
    // In these circumstances you may want to backoff and retry.
    if (message.contains("ConcurrentModificationException")) {
      System.out.println("Retrying because ConcurrentModificationException");
      return true;
    }
```

```
    // If the primary fails over to a new instance, existing connections to the old
 primary will
    // throw a ReadOnlyViolationException. You may want to back and retry.
    if (message.contains("ReadOnlyViolationException")) {
      System.out.println("Retrying because ReadOnlyViolationException");
      return true;
    }

    System.out.println("Not a retriable error");
    return false;
  };
}
```

Utilisation de Python pour se connecter à une instance de base de données Neptune

> **Important**
>
> Le choix de la bonne version du pilote TinkerPop Apache G705 est essentiel pour garantir la compatibilité avec la version de votre moteur Neptune. L'utilisation d'une version incompatible peut entraîner des échecs de connexion ou un comportement inattendu. Pour obtenir des informations détaillées sur la compatibilité des versions, consultezAccès au graphe Neptune avec Gremlin.

La section suivante vous accompagne dans l'exécution d'un exemple Python qui se connecte à une instance de base de données Amazon Neptune et effectue une traversée Gremlin.

Vous devez suivre ces instructions à partir d'une instance Amazon EC2 dans le même cloud privé virtuel (VPC) (VPC) que l'instance de base de données Neptune.

Avant de commencer, vous devez exécuter les actions suivantes :

- Téléchargez et installez Python 3.6 ou version ultérieure depuis le Python.org site Web.
- Vérifiez que vous avez installé pip. Si vous n'avez pas pip ou que vous n'en êtes pas certain, voir Do I need to install pip? (Dois-je installer pip? dans la documentation pip.
- Si votre installation Python n'est pas déjà définie, téléchargez `futures` comme suit : `pip install futures`

Pour se connecter à Neptune à l'aide de Python

1. Entrez ce qui suit pour installer le package `gremlinpython` :

```
pip install --user gremlinpython
```

2. Créez un fichier nommé `gremlinexample.py` et ouvrez-le dans un éditeur de texte.
3. Copiez ce qui suit dans le fichier `gremlinexample.py`. Remplacez *`your-neptune-endpoint`* par l'adresse de votre cluster de base de données Neptune et *`your-neptune-port`* par le port de votre cluster de base de données Neptune (par défaut : 8182).

   Consultez la section Connexion aux points de terminaison Amazon Neptune pour découvrir comment trouver l'adresse de votre instance de base de données Neptune.

   L'exemple ci-dessous montre comment se connecter à Garmlin Python.

```
import boto3
import os
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection
from gremlin_python.process.anonymous_traversal import traversal

database_url = "wss://your-neptune-endpoint:your-neptune-port/gremlin"

remoteConn = DriverRemoteConnection(database_url, "g")

g = traversal().withRemote(remoteConn)

print(g.inject(1).toList())
remoteConn.close()
```

4. Pour exécuter l'exemple, entrez la commande suivante :

```
python gremlinexample.py
```

   La requête Gremlin à la fin de cet exemple renvoie les vertex (`g.V().limit(2)`) dans une liste. Cette liste est ensuite imprimée avec la fonction standard Python `print`.

> **Note**
>
> La partie finale de la requête Gremlin, `toList()`, est obligatoire pour soumettre la traversée au serveur à des fins d'évaluation. Si vous n'incluez pas cette méthode ou une autre méthode équivalente, la requête ne sera pas soumise à l'instance de base de données Neptune.

Voici des exemples de méthodes qui soumettent la requête à l'instance de base de données Neptune :

- `toList()`
- `toSet()`
- `next()`
- `iterate()`

Ces étapes du terminal se comportent différemment en mode script et en mode bytecode. Pour la liste canonique des étapes du terminal et des détails sur la manière dont elles affectent les transactions, voirTestez le code Gremlin dans le contexte dans lequel vous allez le déployer.

`iterate()`À utiliser lorsque vous n'avez pas besoin des résultats de vos requêtes (par exemple des mutations) car cela permet de réduire les coûts de sérialisation.

L'exemple précédent renvoie les deux premier vertex du graphe en utilisant la traversée `g.V().limit(2).toList()`. Pour interroger quelque chose d'autre, remplacez la traversée par une autre traversée Gremlin avec l'une des méthodes de fin appropriées.

Authentification IAM

Neptune prend en charge l'authentification IAM pour contrôler l'accès à votre cluster de base de données. Si l'authentification IAM est activée, vous devez utiliser la signature Signature version 4 pour authentifier vos demandes. Pour obtenir des instructions détaillées et des exemples de code pour la connexion à partir d'un client Python, consultezConnexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Gremlin Python.

Utilisation de .NET pour se connecter à une instance de base de données Neptune

> **Important**
>
> Le choix de la bonne version du pilote TinkerPop Apache G705 est essentiel pour garantir la compatibilité avec la version de votre moteur Neptune. L'utilisation d'une version incompatible peut entraîner des échecs de connexion ou un comportement inattendu. Pour obtenir des informations détaillées sur la compatibilité des versions, consultezAccès au graphe Neptune avec Gremlin.

La section suivante contient un exemple de code écrit en C# qui se connecte à une instance de base de données Neptune et effectue une traversée Gremlin.

Les connexions à Amazon Neptune doivent provenir d'une instance Amazon EC2 située dans le même cloud privé virtuel que votre instance de base de données Neptune. Cet exemple de code a été testé sur une instance Amazon EC2 exécutant Ubuntu.

Avant de commencer, vous devez exécuter les actions suivantes :

- Installez .NET sur l'instance Amazon EC2. Pour obtenir des instructions sur la façon d'installer .NET sur plusieurs systèmes d'exploitation, notamment Windows, Linux et macOS, consultez Mise en route avec .NET.
- Installez Gremlin.NET en exécutant `dotnet add package gremlin.net` votre package. Pour plus d'informations, consultez Gremlin.NETla TinkerPop documentation.

Pour vous connecter à Neptune à l'aide de Gremlin.NET

1. Crée un projet .NET.

   ```
   dotnet new console -o gremlinExample
   ```

2. Modifiez les répertoires vers le nouveau répertoire de projet.

   ```
   cd gremlinExample
   ```

3. Copiez ce qui suit dans le fichier `Program.cs`. Remplacez *`your-neptune-endpoint`* par l'adresse de votre instance de base de données Neptune.

   Consultez la section Connexion aux points de terminaison Amazon Neptune pour découvrir comment trouver l'adresse de votre instance de base de données Neptune.

```
using System;
using System.Threading.Tasks;
using System.Collections.Generic;
using Gremlin.Net;
using Gremlin.Net.Driver;
using Gremlin.Net.Driver.Remote;
using Gremlin.Net.Structure;
using static Gremlin.Net.Process.Traversal.AnonymousTraversalSource;
namespace gremlinExample
{
  class Program
  {
    static void Main(string[] args)
    {
      try
      {
        var endpoint = "your-neptune-endpoint";
        // This uses the default Neptune and Gremlin port, 8182
        var gremlinServer = new GremlinServer(endpoint, 8182, enableSsl: true );
        var gremlinClient = new GremlinClient(gremlinServer);
        var remoteConnection = new DriverRemoteConnection(gremlinClient, "g");
        var g = Traversal().WithRemote(remoteConnection);
        g.AddV("Person").Property("Name", "Justin").Iterate();
        g.AddV("Custom Label").Property("name", "Custom id vertex 1").Iterate();
        g.AddV("Custom Label").Property("name", "Custom id vertex 2").Iterate();
        var output = g.V().Limit<Vertex>(3).ToList();
        foreach(var item in output) {
            Console.WriteLine(item);
        }
      }
      catch (Exception e)
      {
          Console.WriteLine("{0}", e);
      }
    }
  }
```

```
}
```

4. Pour exécuter l'exemple, entrez la commande suivante :

```
dotnet run
```

La requête Gremlin à la fin de cet exemple renvoie le nombre d'un seul vertex à des fins de test. Elle est ensuite imprimée sur la console.

> **Note**
>
> La partie finale de la requête Gremlin, `Next()`, est obligatoire pour soumettre la traversée au serveur à des fins d'évaluation. Si vous n'incluez pas cette méthode ou une autre méthode équivalente, la requête ne sera pas soumise à l'instance de base de données Neptune.

Voici des exemples de méthodes qui soumettent la requête à l'instance de base de données Neptune :

- `ToList()`
- `ToSet()`
- `Next()`
- `Iterate()`

Ces étapes du terminal se comportent différemment en mode script et en mode bytecode. Pour la liste canonique des étapes du terminal et des détails sur la manière dont elles affectent les transactions, voirTestez le code Gremlin dans le contexte dans lequel vous allez le déployer.

`Iterate()`À utiliser lorsque vous n'avez pas besoin des résultats de vos requêtes (par exemple des mutations) car cela permet de réduire les coûts de sérialisation.

L'exemple précédent renvoie une liste à l'aide de la traversée `g.V().Limit(3).ToList()`. Pour interroger quelque chose d'autre, remplacez la traversée par une autre traversée Gremlin avec l'une des méthodes de fin appropriées.

Authentification IAM

Neptune prend en charge l'authentification IAM pour contrôler l'accès à votre cluster de base de données. Si l'authentification IAM est activée, vous devez utiliser la signature Signature version 4 pour authentifier vos demandes. Pour obtenir des instructions détaillées et des exemples de code pour la connexion à partir d'un client .NET, consultezConnexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Gremlin .NET.

Utilisation Node.js pour se connecter à une instance de base de données Neptune

> **Important**
>
> Le choix de la bonne version du pilote TinkerPop Apache G705 est essentiel pour garantir la compatibilité avec la version de votre moteur Neptune. L'utilisation d'une version incompatible peut entraîner des échecs de connexion ou un comportement inattendu. Pour obtenir des informations détaillées sur la compatibilité des versions, consultezAccès au graphe Neptune avec Gremlin.

La section suivante explique comment exécuter un Node.js exemple qui se connecte à une instance de base de données Amazon Neptune et effectue une traversée Gkremlin.

Vous devez suivre ces instructions à partir d'une instance Amazon EC2 dans le même cloud privé virtuel (VPC) (VPC) que l'instance de base de données Neptune.

Avant de commencer, vous devez exécuter les actions suivantes :

- Vérifiez que Node.js la version 8.11 ou supérieure est installée. Si ce n'est pas le cas, téléchargez-le et installez-le Node.js depuis le Nodejs.org site Web.

Pour vous connecter à Neptune à l'aide de Node.js

1. Entrez ce qui suit pour installer le package `gremlin-javascript` :

   ```
   npm install gremlin
   ```

2. Créez un fichier nommé `gremlinexample.js` et ouvrez-le dans un éditeur de texte.

3. Copiez ce qui suit dans le fichier `gremlinexample.js`. Remplacez *`your-neptune-endpoint`* par l'adresse de votre instance de base de données Neptune.

   Consultez la section Connexion aux points de terminaison Amazon Neptune pour découvrir comment trouver l'adresse de votre instance de base de données Neptune.

```
const gremlin = require('gremlin');
const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;
const Graph = gremlin.structure.Graph;

dc = new DriverRemoteConnection('wss://your-neptune-endpoint:8182/gremlin',{});

const graph = new Graph();
const g = graph.traversal().withRemote(dc);

g.V().limit(1).count().next().
    then(data => {
        console.log(data);
        dc.close();
    }).catch(error => {
        console.log('ERROR', error);
        dc.close();
    });
```

4. Pour exécuter l'exemple, entrez la commande suivante :

```
node gremlinexample.js
```

L'exemple précédent renvoie le nombre d'un seul sommet dans le graphique en utilisant la traversée `g.V().limit(1).count().next()`. Pour interroger quelque chose d'autre, remplacez la traversée par une autre traversée Gremlin avec l'une des méthodes de fin appropriées.

> **Note**
>
> La partie finale de la requête Gremlin, `next()`, est obligatoire pour soumettre la traversée au serveur à des fins d'évaluation. Si vous n'incluez pas cette méthode ou une autre méthode équivalente, la requête ne sera pas soumise à l'instance de base de données Neptune.

Voici des exemples de méthodes qui soumettent la requête à l'instance de base de données Neptune :

- `toList()`
- `toSet()`
- `next()`
- `iterate()`

Ces étapes du terminal se comportent différemment en mode script et en mode bytecode. Pour la liste canonique des étapes du terminal et des détails sur la manière dont elles affectent les transactions, voirTestez le code Gremlin dans le contexte dans lequel vous allez le déployer.

`iterate()`À utiliser lorsque vous n'avez pas besoin des résultats de vos requêtes (par exemple des mutations) car cela permet de réduire les coûts de sérialisation.

> **Important**
>
> Il s'agit d'un Node.js exemple autonome. Si vous envisagez d'exécuter un code de ce type dans une AWS Lambda fonction, consultez Exemples de fonctions Lambda pour plus de détails sur l'utilisation JavaScript efficace dans une fonction Neptune Lambda.

Authentification IAM

Neptune prend en charge l'authentification IAM pour contrôler l'accès à votre cluster de base de données. Si l'authentification IAM est activée, vous devez utiliser la signature Signature version 4 pour authentifier vos demandes. Pour obtenir des instructions détaillées et des exemples de code pour la connexion depuis un JavaScript client, consultezConnexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Gremlin JavaScript.

Utilisation de Go pour se connecter à une instance de base de données Neptune

> **Important**
>
> Le choix de la bonne version du pilote TinkerPop Apache G705 est essentiel pour garantir la compatibilité avec la version de votre moteur Neptune. L'utilisation d'une version incompatible peut entraîner des échecs de connexion ou un comportement inattendu. Pour obtenir des

informations détaillées sur la compatibilité des versions, consultezAccès au graphe Neptune avec Gremlin.

> **Note**
>
> Les versions 3.5.x de gremlingo sont rétrocompatibles avec les versions 3.4.x tant que vous n'utilisez que les fonctionnalités TinkerPop 3.4.x dans les requêtes Gkremlin que vous écrivez.

La section suivante vous accompagne dans l'exécution d'un exemple Go qui se connecte à une instance de base de données Amazon Neptune et effectue une traversée Gremlin.

Vous devez suivre ces instructions à partir d'une instance Amazon EC2 dans le même cloud privé virtuel (VPC) (VPC) que l'instance de base de données Neptune.

Avant de commencer, vous devez exécuter les actions suivantes :

- Téléchargez et installez Go 1.17 ou version ultérieure depuis le site web go.dev.

Pour se connecter à Neptune à l'aide de Go

1. À partir d'un répertoire vide, initialisez un nouveau module Go :

   ```
   go mod init example.com/gremlinExample
   ```

2. Ajoutez gremlin-go comme dépendance du nouveau module :

   ```
   go get github.com/apache/tinkerpop/gremlin-go/v3/driver
   ```

3. Créez un fichier nommé `gremlinExample.go` et ouvrez-le dans un éditeur de texte.
4. Copiez ce qui suit dans le fichier `gremlinExample.go`, en remplaçant *`(your neptune endpoint)`* par l'adresse de l'instance de base de données Neptune :

   ```
   package main

   import (
     "fmt"
     gremlingo "github.com/apache/tinkerpop/gremlin-go/v3/driver"
   ```

```
)

func main() {
  // Creating the connection to the server.
  driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection("wss://(your
 neptune endpoint):8182/gremlin",
    func(settings *gremlingo.DriverRemoteConnectionSettings) {
      settings.TraversalSource = "g"
    })
  if err != nil {
    fmt.Println(err)
    return
  }
  // Cleanup
  defer driverRemoteConnection.Close()

  // Creating graph traversal
  g := gremlingo.Traversal_().WithRemote(driverRemoteConnection)

  // Perform traversal
  results, err := g.V().Limit(2).ToList()
  if err != nil {
    fmt.Println(err)
    return
  }
  // Print results
  for _, r := range results {
    fmt.Println(r.GetString())
  }
}
```

Note

Le format du certificat Neptune TLS n'est actuellement pas pris en charge sur Go 1.18 ou version supérieure avec macOS et peut générer une erreur 509 lorsque vous essayez d'établir une connexion. Pour les tests locaux, cela peut être ignoré en ajoutant « crypto/tls » aux importations et en modifiant les `DriverRemoteConnection` paramètres comme suit :

```
// Creating the connection to the server.
driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection("wss://
your-neptune-endpoint:8182/gremlin",
```

```
    func(settings *gremlingo.DriverRemoteConnectionSettings) {
        settings.TraversalSource = "g"
        settings.TlsConfig = &tls.Config{InsecureSkipVerify: true}
    })
```

5. Pour exécuter l'exemple, entrez la commande suivante :

```
go run gremlinExample.go
```

La requête Gremlin à la fin de cet exemple renvoie les sommets (`(g.V().Limit(2))`) dans une liste. Cette tranche est ensuite itérée et imprimée avec la fonction `fmt.Println` standard.

> **Note**
>
> La partie finale de la requête Gremlin, `ToList()`, est obligatoire pour soumettre la traversée au serveur à des fins d'évaluation. Si vous n'incluez pas cette méthode ou une autre méthode équivalente, la requête ne sera pas soumise à l'instance de base de données Neptune.

Voici des exemples de méthodes qui soumettent la requête à l'instance de base de données Neptune :

- `ToList()`
- `ToSet()`
- `Next()`
- `Iterate()`

Ces étapes du terminal se comportent différemment en mode script et en mode bytecode. Pour la liste canonique des étapes du terminal et des détails sur la manière dont elles affectent les transactions, voirTestez le code Gremlin dans le contexte dans lequel vous allez le déployer.

`Iterate()`À utiliser lorsque vous n'avez pas besoin des résultats de vos requêtes (par exemple des mutations) car cela permet de réduire les coûts de sérialisation.

L'exemple précédent renvoie les deux premier vertex du graphe en utilisant la traversée `g.V().Limit(2).ToList()`. Pour interroger quelque chose d'autre, remplacez la traversée par une autre traversée Gremlin avec l'une des méthodes de fin appropriées.

Authentification IAM

Neptune prend en charge l'authentification IAM pour contrôler l'accès à votre cluster de base de données. Si l'authentification IAM est activée, vous devez utiliser la signature Signature version 4 pour authentifier vos demandes. Pour obtenir des instructions détaillées et des exemples de code pour la connexion à partir d'un client Go, consultezConnexion aux bases de données Amazon Neptune à l'aide de l'authentification IAM avec Gremlin Go.

Utilisation de AWS SDK pour exécuter des requêtes G705

Avec le AWS SDK, vous pouvez exécuter des requêtes G705 sur votre graphe Neptune à l'aide du langage de programmation de votre choix. Le SDK de l'API de données Neptune (nom du service`neptunedata`) fournit l'ExecuteGremlinQueryaction permettant de soumettre des requêtes Gremlin.

Vous devez exécuter ces exemples depuis une instance Amazon EC2 dans le même cloud privé virtuel (VPC) que votre cluster de base de données Neptune, ou depuis un emplacement disposant d'une connectivité réseau avec le point de terminaison de votre cluster.

Vous trouverez ci-dessous des liens directs vers la documentation de référence `neptunedata` de l'API pour le service dans chaque langue du SDK :

Langage de programmation	Référence de l'API neptunedata
C++	https://sdk.amazonaws.com/cpp/api/LATEST/aws-cpp-sdk-neptunedata/html/annotated.html
Go	https://docs.aws.amazon.com/sdk-for-go/api/service/neptunedata/
Java	https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/services/neptunedata/package-summary.html
JavaScript	https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/Package/-aws-sdk-client-neptunedata/

Langage de programmation	Référence de l'API neptunedata
Kotlin	https://sdk.amazonaws.com/kotlin/api/latest/neptunedata/index.html
.NET	https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Neptunedata/NNeptunedata.html
PHP	https://docs.aws.amazon.com/aws-sdk-php/v3/api/namespace-Aws.Neptunedata.html
Python	https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/neptunedata.html
Ruby	https://docs.aws.amazon.com/sdk-for-ruby/v3/api/Aws/Neptunedata.html
Rust	https://crates.io/crates/aws-sdk-neptunedata
INTERFACE DE LIGNE DE COMMANDE (CLI)	https://docs.aws.amazon.com/cli/latest/reference/neptunedata/

Gremlin AWS Exemples de SDK

Les exemples suivants montrent comment configurer un `neptunedata` client, exécuter une requête Gkremlin et imprimer les résultats. Remplacez *`YOUR_NEPTUNE_HOST`* et *`YOUR_NEPTUNE_PORT`* par le point de terminaison et le port de votre cluster de base de données Neptune.

⚠ Client-side configuration du délai d'expiration et des nouvelles tentatives

Le délai d'attente du client SDK contrôle le temps d'attente d'une réponse par le client. Il ne contrôle pas la durée d'exécution de la requête sur le serveur. Si le client expire avant la fin du serveur, la requête peut continuer à s'exécuter sur Neptune alors que le client n'a aucun moyen de récupérer les résultats.

Nous vous recommandons de définir le délai de lecture côté client sur `0` (aucun délai d'attente) ou sur une valeur supérieure d'au moins quelques secondes au paramètre

neptune_query_timeout côté serveur sur votre cluster de base de données Neptune. Cela permet à Neptune de contrôler l'expiration du délai d'expiration des requêtes.
Nous recommandons également de fixer le nombre maximum de tentatives à 1 (aucune tentative). Si le SDK réessaie une requête toujours en cours d'exécution sur le serveur, cela peut entraîner des opérations dupliquées. Cela est particulièrement important pour les requêtes de mutation, où une nouvelle tentative peut entraîner des écritures dupliquées involontaires.

Python

1. Suivez les instructions d'installation pour installer Boto3.
2. Créez un fichier nommé `gremlinExample.py` et collez le code suivant :

```
import boto3
import json
from botocore.config import Config

# Disable the client-side read timeout and retries so that
# Neptune's server-side neptune_query_timeout controls query duration.
client = boto3.client(
    'neptunedata',
    endpoint_url=f'https://YOUR_NEPTUNE_HOST:YOUR_NEPTUNE_PORT',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

# Use the untyped GraphSON v3 serializer for a cleaner JSON response.
response = client.execute_gremlin_query(
    gremlinQuery='g.V().limit(1)',
    serializer='application/vnd.gremlin-v3.0+json;types=false'
)

print(json.dumps(response['result'], indent=2))
```

3. Exécutez l'exemple : `python gremlinExample.py`

Java

1. Suivez les instructions d'installation pour configurer le AWS SDK for Java.

2. Utilisez le code suivant pour configurer une`NeptunedataClient`, exécuter une requête G705 et imprimer le résultat :

```
import java.net.URI;
import java.time.Duration;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.services.neptunedata.NeptunedataClient;
import
 software.amazon.awssdk.services.neptunedata.model.ExecuteGremlinQueryRequest;
import
 software.amazon.awssdk.services.neptunedata.model.ExecuteGremlinQueryResponse;

// Disable the client-side timeout and retries so that
// Neptune's server-side neptune_query_timeout controls query duration.
NeptunedataClient client = NeptunedataClient.builder()
    .endpointOverride(URI.create("https://YOUR_NEPTUNE_HOST:YOUR_NEPTUNE_PORT"))
    .overrideConfiguration(ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ZERO)
        .retryPolicy(RetryPolicy.none())
        .build())
    .build();

// Use the untyped GraphSON v3 serializer for a cleaner JSON response.
ExecuteGremlinQueryRequest request = ExecuteGremlinQueryRequest.builder()
    .gremlinQuery("g.V().limit(1)")
    .serializer("application/vnd.gremlin-v3.0+json;types=false")
    .build();

ExecuteGremlinQueryResponse response = client.executeGremlinQuery(request);

System.out.println(response.result().toString());
```

JavaScript

1. Suivez les instructions d'installation pour configurer le AWS SDK pour JavaScript. Installez le package client neptunedata :. `npm install @aws-sdk/client-neptunedata`
2. Créez un fichier nommé `gremlinExample.js` et collez le code suivant :

```
import { NeptunedataClient, ExecuteGremlinQueryCommand } from "@aws-sdk/client-
neptunedata";
```

```
import { NodeHttpHandler } from "@smithy/node-http-handler";

const config = {
    endpoint: "https://YOUR_NEPTUNE_HOST:YOUR_NEPTUNE_PORT",
    // Disable the client-side request timeout so that
    // Neptune's server-side neptune_query_timeout controls query duration.
    requestHandler: new NodeHttpHandler({
        requestTimeout: 0
    }),
    maxAttempts: 1
};

const client = new NeptunedataClient(config);

// Use the untyped GraphSON v3 serializer for a cleaner JSON response.
const input = {
    gremlinQuery: "g.V().limit(1)",
    serializer: "application/vnd.gremlin-v3.0+json;types=false"
};

const command = new ExecuteGremlinQueryCommand(input);
const response = await client.send(command);

console.log(JSON.stringify(response, null, 2));
```

3. Exécutez l'exemple : `node gremlinExample.js`

Indicateurs de requête Gremlin

Vous pouvez utiliser des indicateurs de requête afin de spécifier des stratégies d'optimisation et d'évaluation pour une requête Gremlin particulière dans Amazon Neptune.

La plupart des indications de requête sont spécifiées en ajoutant une `withSideEffect` étape à la requête avec la syntaxe suivante.

```
g.withSideEffect(hint, value)
```

- hint : identifie le type d'indicateur à appliquer.
- value : détermine le comportement de l'aspect du système pris en compte.

Par exemple, le code suivant montre comment inclure un indicateur `repeatMode` dans une traversée Gremlin.

> **Note**
>
> Les indications de requête utilisées `withSideEffect` sont toujours préfixées par`Neptune#`.

```
g.withSideEffect('Neptune#repeatMode',
 'DFS').V("3").repeat(out()).times(10).limit(1).path()
```

La requête précédente demande au moteur Neptune de parcourir le graphe en profondeur en premier (`DFS`) au lieu du mode par défaut, dans la largeur en premier (`BFS`).

> **Important**
>
> Certains indices de requête utilisent l'`with`étape au lieu de`withSideEffect`. Les deux ne sont pas interchangeables. Chaque section d'indices de requête de cette page indique la syntaxe correcte à utiliser.

Les sections suivantes fournissent de plus amples informations sur les indicateurs de requête disponibles et leur utilisation.

Rubriques

- Indicateur de requête Gremlin repeatMode
- Indicateur de requête Gremlin noReordering
- Indicateur de requête Gremlin typePromotion
- Indicateur de requête Gremlin useDFE
- Conseil de TimeoutBehavior requête d'évaluation Garmlin
- Indicateurs de requête Gremlin pour l'utilisation du cache de résultats

Indicateur de requête Gremlin repeatMode

L'indicateur de requête Neptune `repeatMode` indique la façon dont le moteur Neptune évalue l'étape `repeat()` dans une traversée Gremlin : largeur en premier, profondeur en premier ou profondeur en premier par blocs.

Le mode d'évaluation de l'étape `repeat()` est important lorsqu'il est utilisé pour rechercher ou suivre un chemin, au lieu de simplement répéter une étape un nombre limité de fois.

Syntaxe

L'indicateur de requête `repeatMode` est spécifié en ajoutant une étape `withSideEffect` à la requête.

```
g.withSideEffect('Neptune#repeatMode', 'mode').gremlin-traversal
```

> **Note**
>
> Tous les effets secondaires d'indicateurs de requête Gremlin sont préfixés avec `Neptune#`.

Modes disponibles

- `BFS`

  Breadth-First Rechercher

  Mode d'exécution par défaut pour l'étape `repeat()`. Ce mode permet d'obtenir tous les nœuds de même niveau avant d'aller plus en profondeur dans le chemin.

  Cette version est gourmande en mémoire et les frontières peuvent être très étendues. Le risque que la requête manque de mémoire et soit annulée par le moteur Neptune est plus élevé. Ce mode correspond au plus près aux autres implémentations Gremlin.
- `DFS`

  Depth-First Rechercher

  Suit chaque chemin jusqu'à la profondeur maximale avant de passer à la solution suivante.

  Ce mode utilise moins de mémoire. Il peut offrir de meilleures performances dans des situations comme la recherche d'un chemin unique à partir d'un point de départ vers plusieurs tronçons.
- `CHUNKED_DFS`

  Recherche fragmentée Depth-First

Approche hybride qui explore le graphe en profondeur en premier par blocs de 1 000 nœuds, plutôt que 1 nœud (DFS) ou tous les nœuds (BFS).

Le moteur Neptune extrait jusqu'à 1 000 nœuds à chaque niveau avant de suivre le chemin plus en profondeur.

Cela constitue un bon compromis entre la vitesse et l'utilisation de la mémoire.

Cette approche s'avère également utile si vous souhaitez utiliser la mode BFS, mais la que requête utilise trop de mémoire.

Exemple

La section suivante décrit l'effet du mode de répétition sur un parcours Gremlin.

Dans Neptune, le mode par défaut pour l'étape `repeat()` consiste à effectuer une stratégie d'exécution dans la largeur en premier (BFS) pour toutes les traversées.

Dans la plupart des cas, l' TinkerGraph implémentation utilise la même stratégie d'exécution, mais dans certains cas, elle modifie l'exécution d'une traversée.

Par exemple, l' TinkerGraph implémentation modifie la requête suivante.

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

L'étape `repeat()` de ce parcours est « déroulée » dans le parcours suivant, ce qui se traduit par une stratégie en profondeur en premier (DFS).

```
g.V(<id>).out().out().out().out().out().out().out().out().out().out().limit(1).path()
```

> **Important**
>
> Le moteur de requête Neptune ne procède pas ainsi automatiquement.

Breadth-first (BFS) est la stratégie d'exécution par défaut, similaire TinkerGraph dans la plupart des cas. Cependant, il y a certains cas où les stratégies en profondeur en premier (DFS) sont préférables.

BFS (valeur par défaut)

Breadth-first (BFS) est la stratégie d'exécution par défaut de l'`repeat()`opérateur.

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

Le moteur Neptune explore entièrement les premières frontières à neuf tronçons avant de rechercher une solution à partir du dixième tronçon. Cette approche est efficace dans de nombreux cas, notamment dans celui d'une requête au chemin le plus court.

Cependant, dans l'exemple précédent, le parcours serait beaucoup plus rapide à l'aide du mode profondeur en premier (DFS) pour l'opérateur `repeat()`.

DFS

La requête suivante utilise le mode DFS (profondeur en premier) pour l'opérateur `repeat()`.

```
g.withSideEffect("Neptune#repeatMode", "DFS").V("3").repeat(out()).times(10).limit(1)
```

Ce mode suit chaque solution jusqu'à la profondeur maximale avant d'explorer la solution suivante.

Indicateur de requête Gremlin noReordering

Lorsque vous soumettez une traversée Gremlin, le moteur de requête Neptune étudie sa structure et réorganise les parties de la requête, en tentant de réduire la quantité de travail nécessaire pour l'évaluation et les temps de réponse de la requête. Par exemple, un parcours avec plusieurs contraintes, telles que plusieurs étapes `has()`, n'est généralement pas évalué dans l'ordre donné. Au lieu de cela, il est réorganisé une fois que la requête a été vérifiée avec une analyse statique.

Le moteur de requête Neptune essaie d'identifier la contrainte la plus sélective et exécute celle-ci en premier. Cela se traduit souvent par de meilleures performances, mais l'ordre dans lequel Neptune choisit d'évaluer la requête peut ne pas toujours être optimal.

Si vous connaissez les caractéristiques exactes des données et que vous souhaitez imposer manuellement l'ordre d'exécution de la requête, vous pouvez utiliser l'indicateur de requête Neptune `noReordering` pour demander que la requête soit évaluée dans l'ordre donné.

Syntaxe

L'indicateur de requête `noReordering` est spécifié en ajoutant une étape `withSideEffect` à la requête.

```
g.withSideEffect('Neptune#noReordering', true or false).gremlin-traversal
```

> **Note**
>
> Tous les effets secondaires d'indicateurs de requête Gremlin sont préfixés avec `Neptune#`.

Valeurs disponibles

- `true`
- `false`

Indicateur de requête Gremlin typePromotion

Lorsque vous soumettez une traversée Gremlin qui filtre une valeur ou une plage numérique, le moteur de requête Neptune doit normalement utiliser la promotion de type lorsqu'il exécute la requête. Autrement dit, il doit examiner les valeurs de tous les types susceptibles de contenir la valeur sur laquelle porte le filtre.

Par exemple, si vous filtrez les valeurs égales à 55, le moteur doit rechercher les entiers égaux à 55, les entiers longs égaux à 55L, les nombres flottants égaux à 55,0, etc. Chaque promotion de type implique une recherche supplémentaire au niveau du stockage, ce qui peut entraîner un délai étonnamment long pour terminer une requête apparemment simple.

Supposons que vous recherchiez tous les sommets dont la propriété correspondant à l'âge du client est supérieure à 5 :

```
g.V().has('customerAge', gt(5))
```

Pour exécuter cette traversée de manière approfondie, Neptune doit développer la requête afin d'examiner tous les types numériques vers lesquels la valeur que vous recherchez pourrait être promue. Dans ce cas, le filtre `gt` doit être appliqué pour tout entier supérieur à 5, tout long supérieur à 5L, tout flottant supérieur à 5,0 et tout double supérieur à 5,0. Étant donné que chacune de ces promotions nécessite une recherche supplémentaire sur le stockage, vous verrez plusieurs filtres par filtre numérique lorsque vous exécuterez l'API Gremlin `profile` pour cette requête, et son exécution prendra beaucoup plus de temps que prévu.

Souvent, la promotion de type n'est pas nécessaire, car vous savez déjà que vous n'avez besoin de trouver que les valeurs d'un type spécifique. Dans ce cas, vous pouvez accélérer considérablement les requêtes en utilisant l'indicateur de requête `typePromotion` afin de désactiver la promotion de type.

Syntaxe

L'indicateur de requête `typePromotion` est spécifié en ajoutant une étape `withSideEffect` à la requête.

```
g.withSideEffect('Neptune#typePromotion', true or false).gremlin-traversal
```

> **Note**
>
> Tous les effets secondaires d'indicateurs de requête Gremlin sont préfixés avec `Neptune#`.

Valeurs disponibles

- `true`
- `false`

Pour désactiver la promotion de type pour la requête ci-dessus, vous devez utiliser :

```
g.withSideEffect('Neptune#typePromotion', false).V().has('customerAge', gt(5))
```

Indicateur de requête Gremlin useDFE

Utilisez cet indicateur de requête afin d'activer l'utilisation du DFE pour exécuter la requête. Par défaut, Neptune n'utilise pas le DFE sans que cet indicateur de requête ne soit défini sur `true`, car le paramètre d'instance neptune_dfe_query_engine est défini par défaut sur `viaQueryHint`. Si vous définissez ce paramètre d'instance sur `enabled`, le moteur DFE est utilisé pour toutes les requêtes, à l'exception de celles dont l'indicateur de requête `useDFE` est défini sur `false`.

Exemple d'activation du DFE pour une requête :

```
g.withSideEffect('Neptune#useDFE', true).V().out()
```

Conseil de TimeoutBehavior requête d'**`évaluation`** Garmlin

Par défaut, lorsqu'une requête Gremlin dépasse le délai d'expiration de la requête, Neptune annule la requête et renvoie une erreur. L'indice de `evaluationTimeoutBehavior` requête modifie ce comportement afin que Neptune renvoie tous les résultats partiels calculés avant l'expiration du délai, au lieu de renvoyer une erreur.

> **Note**
>
> Cet indice de requête utilise l'`with`étape, non`withSideEffect`, et n'utilise pas le `Neptune#` préfixe.

Syntaxe

```
g.with('evaluationTimeoutBehavior', 'value').gremlin-traversal
```

Valeurs disponibles

- `partialResults`— Lorsque la requête expire, Neptune renvoie tous les résultats calculés avant le délai d'expiration.
- `exception`— Lorsque le délai de requête expire, Neptune renvoie une erreur. Il s'agit du comportement de par défaut.

> **Important**
>
> Cet indice de requête ne peut être utilisé qu'avec des requêtes en lecture seule. Si vous utilisez cet indice avec une requête mutante ou si le DFE est activé (via l'indice de `useDFE` requête ou le paramètre d'`neptune_dfe_query_engine`instance), Neptune renvoie une erreur.

Exemple

La requête suivante renvoie des résultats partiels si elle dépasse le délai d'expiration de la requête :

```
g.with('evaluationTimeoutBehavior', 'partialResults').V().out().valueMap()
```

Indicateurs de requête Gremlin pour l'utilisation du cache de résultats

Les indicateurs de requête suivants peuvent être utilisés lorsque le cache des résultats de requête est activé.

Astuce d'**`activation`** de la ResultCache requête par Garmlin

Lorsque l'indicateur de requête `enableResultCache` a une valeur égale à `true`, les résultats de la requête sont renvoyés à partir du cache s'ils ont déjà été mis en cache. Dans le cas contraire, de nouveaux résultats sont renvoyés et mis en cache jusqu'à ce qu'ils soient effacés de celui-ci. Par exemple :

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Plus tard, vous pourrez accéder aux résultats mis en cache en émettant à nouveau exactement la même requête.

Si la valeur de cet indicateur de requête est `false` ou si elle n'est pas présente, les résultats de la requête ne sont pas mis en cache. Toutefois, l'utilisation du paramètre `false` n'efface pas les résultats mis en cache existants. Pour effacer les résultats mis en cache, utilisez l'indicateur `invalidateResultCache` ou `invalidateResultCachekey`.

Astuce d'**`activation`** de la ResultCacheWithTTL requête par Garmlin

L'indicateur de requête `enableResultCacheWithTTL` renvoie également les résultats mis en cache s'il y en a, sans affecter la durée de vie (TTL) des résultats déjà présents dans le cache. S'il n'y a actuellement aucun résultat mis en cache, la requête renvoie de nouveaux résultats et les met en cache pendant la durée de vie (TTL) spécifiée par l'indicateur de requête `enableResultCacheWithTTL`. Cette durée de vie est spécifiée en secondes. Par exemple, la requête suivante spécifie une durée de vie de 60 secondes :

```
g.with('Neptune#enableResultCacheWithTTL', 60)
 .V().has('genre','drama').in('likes')
```

Avant la fin du délai de 60 secondes, vous pouvez utiliser la même requête (ici `g.V().has('genre','drama').in('likes')`) avec l'indicateur de requête `enableResultCache` ou `enableResultCacheWithTTL` pour accéder aux résultats mis en cache.

> **Note**
>
> La durée de vie spécifiée avec `enableResultCacheWithTTL` n'affecte pas les résultats déjà mis en cache.
>
> - Si les résultats ont déjà été mis en cache avec `enableResultCache`, le cache doit d'abord être explicitement vidé avant qu'`enableResultCacheWithTTL` génère de nouveaux résultats et les mette en cache pour le TTL qu'il spécifie.
> - Si les résultats ont déjà été mis en cache à l'aide de l'indicateur `enableResultCachewithTTL`, cette durée de vie précédente, ou TTL, doit d'abord expirer avant qu'`enableResultCacheWithTTL` génère de nouveaux résultats et les mette en cache pour la durée de vie qu'il spécifie.

Une fois la durée de vie écoulée, les résultats mis en cache pour la requête sont effacés, et toute instance ultérieure de la même requête renverra de nouveaux résultats. Si `enableResultCacheWithTTL` est attaché à cette requête ultérieure, les nouveaux résultats sont mis en cache avec le TTL spécifié.

Indice de requête d'**`invalidation par ResultCacheKey`** Garmlin

L'indicateur de requête `invalidateResultCacheKey` peut avoir la valeur `false` ou `true`. Une valeur `true` entraîne l'effacement des résultats mis en cache pour la requête à laquelle `invalidateResultCacheKey` est attaché. Par exemple, dans l'exemple suivant, les résultats mis en cache pour la clé de requête `g.V().has('genre','drama').in('likes')` sont effacés :

```
g.with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

L'exemple de requête ci-dessus n'entraîne pas la mise en cache de ses nouveaux résultats. Vous pouvez inclure `enableResultCache` (ou `enableResultCacheWithTTL`) dans la même requête si vous souhaitez mettre en cache les nouveaux résultats après avoir effacé les résultats existants dans le cache :

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Indice de requête d'**invalidation par ResultCache** Garmlin

L'indicateur de requête `invalidateResultCache` peut avoir la valeur `false` ou `true`. Une valeur `true` entraîne l'effacement de tous les résultats du cache de résultats. Par exemple :

```
g.with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

L'exemple de requête ci-dessus n'entraîne pas la mise en cache de ses résultats. Vous pouvez inclure `enableResultCache` (ou `enableResultCacheWithTTL`) dans la même requête si vous souhaitez mettre en cache les nouveaux résultats après avoir entièrement vidé le cache existant :

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Astuce de ResultsCached requête **Garmlin num**

L'indicateur de requête `numResultsCached` ne peut être utilisé qu'avec les requêtes contenant `iterate()`. Il indique le nombre maximal de résultats à mettre en cache pour la requête à laquelle il est attaché. Notez que les résultats mis en cache lorsque `numResultsCached` est présent ne sont pas renvoyés, mais uniquement mis en cache.

Par exemple, la requête suivante spécifie que jusqu'à 100 de ses résultats doivent être mis en cache, mais aucun de ces résultats mis en cache n'est renvoyé :

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').iterate()
```

Vous pouvez ensuite utiliser une requête comme celle-ci pour récupérer une série de résultats mis en cache (ici, les dix premiers) :

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').range(0, 10)
```

Gremlin : **aucun indice** de CacheExceptions requête

L'indicateur de requête `noCacheExceptions` peut avoir la valeur `false` ou `true`. Une valeur `true` entraîne la suppression de toutes les exceptions liées au cache de résultats. Par exemple :

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#noCacheExceptions', true)
 .V().has('genre','drama').in('likes')
```

Elle supprime notamment l'exception `QueryLimitExceededException`, qui est déclenchée si les résultats d'une requête sont trop volumineux pour tenir dans le cache de résultats.

API de statut des requêtes Gremlin

Vous pouvez répertorier toutes les requêtes Gremlin actives ou obtenir le statut d'une requête spécifique. Le point de terminaison HTTP sous-jacent pour les deux opérations est`https://`*your-neptune-endpoint*`:`*port*`/gremlin/status`.

Répertorier les requêtes Gremlin actives

Pour répertorier toutes les requêtes G705 actives, appelez le point de terminaison sans `queryId` paramètre.

Paramètres de demande

- IncludeWaiting (facultatif) — Si cette valeur est définie sur`TRUE`, la réponse inclut les requêtes en attente en plus des requêtes en cours d'exécution.

Syntaxe de réponse

```
{
  "acceptedQueryCount": integer,
  "runningQueryCount": integer,
  "queries": [
    {
      "queryId": "guid",
      "queryEvalStats": {
        "waited": integer,
        "elapsed": integer,
        "cancelled": boolean
      },
      "queryString": "string"
    }
  ]
}
```

- accepté QueryCount — Le nombre de requêtes acceptées mais non encore terminées, y compris les requêtes dans la file d'attente.
- running QueryCount — Le nombre de requêtes Gremlin en cours d'exécution.
- queries : requêtes Gremlin actuelles.

Exemple

AWS CLI

```
aws neptunedata list-gremlin-queries \
  --endpoint-url https://your-neptune-endpoint:port
```

Pour plus d'informations, consultez list-gremlin-queries dans le manuel de référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.list_gremlin_queries()

print(response)
```

Pour des exemples de AWS SDK dans d'autres langages tels que Java, .NET, etc., consultezAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/gremlin/status \
  --region us-east-1 \
  --service neptune-db
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultez[Utiliser awscurl avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée](#).

curl

```
curl https://your-neptune-endpoint:port/gremlin/status
```

Le résultat suivant montre une seule requête en cours d'exécution.

```
{
  "acceptedQueryCount": 9,
  "runningQueryCount": 1,
  "queries": [
    {
      "queryId": "fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
      "queryEvalStats": {
        "waited": 0,
        "elapsed": 23,
        "cancelled": false
      },
      "queryString": "g.V().out().count()"
    }
  ]
}
```

Obtenir le statut d'une requête Gremlin spécifique

Pour obtenir le statut d'une requête G705 spécifique, indiquez le `queryId` paramètre.

Paramètres de demande

- queryID (obligatoire) — L'ID de la requête Gremlin. Neptune attribue automatiquement cette valeur d'identifiant à chaque requête, ou vous pouvez attribuer votre propre identifiant (voir). [Injection d'un ID personnalisé dans une requête Neptune Gremlin ou SPARQL](#)

Syntaxe de réponse

```
{
  "queryId": "guid",
  "queryString": "string",
  "queryEvalStats": {
    "waited": integer,
    "elapsed": integer,
    "cancelled": boolean,
    "subqueries": document
  }
}
```

- queryID — L'ID de la requête.
- queryString : requête soumise. Celle-ci est tronquée à 1 024 caractères si elle est plus longue que cela.
- requête EvalStats : statistiques relatives à la requête, notamment `waited` (temps d'attente en millisecondes), `elapsed` (temps d'exécution en millisecondes), `cancelled` (si la requête a été annulée) et `subqueries` (nombre de sous-requêtes).

Exemple

AWS CLI

```
aws neptunedata get-gremlin-query-status \
  --endpoint-url https://your-neptune-endpoint:port \
  --query-id "fb34cd3e-f37c-4d12-9cf2-03bb741bf54f"
```

Pour plus d'informations, consultez get-gremlin-query-status dans le manuel de référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
```

```
)

response = client.get_gremlin_query_status(
    queryId='fb34cd3e-f37c-4d12-9cf2-03bb741bf54f'
)

print(response)
```

Pour des exemples de AWS SDK dans d'autres langages tels que Java, .NET, etc., consultezAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/gremlin/status/fb34cd3e-
f37c-4d12-9cf2-03bb741bf54f \
  --region us-east-1 \
  --service neptune-db
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultezUtiliser awscurl avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

curl

```
curl https://your-neptune-endpoint:port/gremlin/status/fb34cd3e-
f37c-4d12-9cf2-03bb741bf54f
```

Voici un exemple de réponse.

```
{
  "queryId": "fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
  "queryString": "g.V().out().count()",
  "queryEvalStats": {
    "waited": 0,
```

```
    "elapsed": 23,
    "cancelled": false
  }
}
```

Annulation de requêtes Gremlin

Pour obtenir le statut des requêtes Gremlin, utilisez HTTP `GET` ou `POST` pour effectuer une requête au point de terminaison `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin/status`.

Paramètres des demandes d'annulation des requêtes Gremlin

- cancelQuery : obligatoire pour l'annulation. Ce paramètre n'a aucune valeur correspondante.
- queryId : ID de la requête Gremlin en cours d'exécution à annuler.

Exemple d'annulation de requêtes Gremlin

Voici un exemple d'annulation d'une requête.

AWS CLI

```
aws neptunedata cancel-gremlin-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --query-id "fb34cd3e-f37c-4d12-9cf2-03bb741bf54f"
```

Pour plus d'informations, consultez cancel-gremlin-query dans le manuel de référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.cancel_gremlin_query(
    queryId='fb34cd3e-f37c-4d12-9cf2-03bb741bf54f'
```

```
)

print(response)
```

Pour des exemples de AWS SDK dans d'autres langages tels que Java, .NET, etc., consultezAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/gremlin/status \
  --region us-east-1 \
  --service neptune-db \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=fb34cd3e-f37c-4d12-9cf2-03bb741bf54f"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultezUtiliser `awscurl` avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

curl

```
curl https://your-neptune-endpoint:port/gremlin/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=fb34cd3e-f37c-4d12-9cf2-03bb741bf54f"
```

L'annulation réussie renvoie le code HTTP `200` OK.

Prise en charge des sessions basées sur des scripts Gremlin

Vous pouvez utiliser des sessions Gremlin avec des transactions implicites dans Amazon Neptune. Pour plus d'informations sur les sessions Gremlin, consultez la section Considering Sessions dans la TinkerPop documentation Apache. Les sections ci-dessous décrivent comment utiliser les sessions Gremlin avec Java.

> **Important**
>
> Actuellement, Neptune peut conserver une session basée sur des scripts ouverte jusqu'à 10 minutes. Si vous ne fermez pas votre session avant cela, la session expire et tout ce qu'elle contient est annulé.

Rubriques

- Sessions Gremlin dans la console Gremlin
- Sessions Gremlin dans la variante du langage Gremlin

Sessions Gremlin dans la console Gremlin

Si vous créez une connexion à distance sur la console Gremlin sans le paramètre `session`, la connexion à distance est créée en mode sans session. Dans ce mode, chaque demande envoyée au serveur est traitée comme une transaction complète en elle-même, et aucun état n'est enregistré entre les demandes. Si une demande échoue, seule cette demande est annulée.

Si vous créez une connexion à distance qui utilise le paramètre `session`, vous créez une session basée sur des scripts qui dure jusqu'à ce que vous fermiez la connexion distante. Chaque session est identifiée par un UUID unique que la console génère et renvoie.

Voici un exemple d'appel de console qui crée une session. Une fois les requêtes soumises, un autre appel ferme la session et valide les requêtes.

> **Note**
>
> Le client Gremlin doit toujours être fermé pour libérer des ressources côté serveur.

```
gremlin> :remote connect tinkerpop.server conf/neptune-remote.yaml session
  . . .
  . . .
gremlin> :remote close
```

Pour plus d'informations et des exemples, consultez la section Sessions de la TinkerPop documentation.

Toutes les requêtes que vous exécutez au cours d'une session forment une seule transaction qui n'est pas validée tant que toutes les requêtes n'ont pas abouti et que vous n'avez pas fermé la connexion à distance. Si une requête échoue, ou si vous n'avez pas fermé la connexion avant la fin de la durée de vie de session prise en charge par Neptune, la transaction de session n'est pas validée, et toutes les requêtes qu'elle contient sont annulées.

Sessions Gremlin dans la variante du langage Gremlin

Dans la variante du langage Gremlin (GLV), vous devez créer un objet `SessionedClient` pour émettre plusieurs requêtes en une seule transaction, comme dans l'exemple suivant.

```
try {                              // line 1
  Cluster cluster = Cluster.open();                    // line 2
  Client client = cluster.connect("sessionName");      // line 3
   ...
   ...
} finally {
  // Always close. If there are no errors, the transaction is committed; otherwise,
 it's rolled back.
  client.close();
}
```

La ligne 3 de l'exemple précédent crée l'objet `SessionedClient` selon les options de configuration définies pour le cluster en question. La *`sessionName`* chaîne que vous transmettez à la méthode de connexion devient le nom unique de la session. Pour éviter les collisions, utilisez un UUID pour le nom.

Le client démarre une transaction de session lorsqu'il est initialisé. Toutes les requêtes que vous exécutez au cours du formulaire de session ne sont validées que lorsque vous appelez `client.close( )`. Là encore, si une seule requête échoue ou si vous n'avez pas fermé la connexion avant la fin de la durée de vie de session prise en charge par Neptune, la transaction de session échoue, et toutes les requêtes qu'elle contient sont annulées.

> **Note**
>
> Le client Gremlin doit toujours être fermé pour libérer des ressources côté serveur.

```
GraphTraversalSource g = traversal().withRemote(conn);
```

```
Transaction tx = g.tx();

// Spawn a GraphTraversalSource from the Transaction.
// Traversals spawned from gtx are executed within a single transaction.
GraphTraversalSource gtx = tx.begin();
try {
  gtx.addV('person').iterate();
  gtx.addV('software').iterate();

  tx.commit();
} finally {
    if (tx.isOpen()) {
        tx.rollback();
    }
}
```

Transactions Gremlin dans Neptune

Il existe plusieurs contextes dans lesquels les transactions Gremlin sont exécutées. Lorsque vous travaillez avec Gremlin, il est important de comprendre le contexte dans lequel vous évoluez et quelles en sont les implications :

- **`Script-based`** : les demandes sont effectuées à l'aide de chaînes Gremlin basées sur du texte, comme ceci :
  - Avec le pilote Java et `Client.submit(`*`string`*`)`.
  - Avec la console Gremlin et `:remote connect`.
  - Avec l'API HTTP.
- **`Bytecode-based`** : les demandes sont effectuées à l'aide du bytecode Gremlin sérialisé typique des variantes du langage Gremlin (GLV).

  Par exemple, avec le pilote Java, `g = traversal().withRemote(...)`.

Pour l'un ou l'autre des contextes ci-dessus, il existe un contexte supplémentaire dans lequel la demande est envoyée sans session ou liée à une session.

> **Note**
>
> Les transactions Gremlin doivent toujours être validées ou annulées, afin que les ressources côté serveur puissent être libérées. En cas d'erreur lors de la transaction, il est important de réessayer l'ensemble de la transaction et pas uniquement la demande qui a échoué.

Demandes sans session

En l'absence de session, une demande équivaut à une transaction unique.

Pour les scripts, cela implique qu'une ou plusieurs déclarations Gremlin envoyées dans le cadre d'une seule demande sont validées ou annulées en tant que transaction unique. Par exemple :

```
Cluster cluster = Cluster.open();
Client client = cluster.connect(); // sessionless
// 3 vertex additions in one request/transaction:
client.submit("g.addV();g.addV();g.addV()").all().get();
```

Pour le bytecode, une demande sans session est effectuée pour chaque traversée générée et exécutée à partir de g :

```
GraphTraversalSource g = traversal().withRemote(...);

// 3 vertex additions in three individual requests/transactions:
g.addV().iterate();
g.addV().iterate();
g.addV().iterate();

// 3 vertex additions in one single request/transaction:
g.addV().addV().addV().iterate();
```

Demandes liées à une session

Lorsqu'elles sont liées à une session, plusieurs demandes peuvent être appliquées dans le contexte d'une seule transaction.

Pour les scripts, cela implique qu'il n'est pas nécessaire de concaténer toutes les opérations du graphe en une seule valeur de chaîne intégrée :

```
Cluster cluster = Cluster.open();
```

```
Client client = cluster.connect(sessionName); // session
try {
    // 3 vertex additions in one request/transaction:
    client.submit("g.addV();g.addV();g.addV()").all().get();
} finally {
    client.close();
}

try {
    // 3 vertex additions in three requests, but one transaction:
    client.submit("g.addV()").all().get(); // starts a new transaction with the same
 sessionName
    client.submit("g.addV()").all().get();
    client.submit("g.addV()").all().get();
} finally {
    client.close();
}
```

Pour les sessions basées sur des scripts, la fermeture du client `client.close()` valide la transaction. Aucune commande de restauration explicite n'est disponible dans les sessions basées sur des scripts. Pour forcer une annulation, vous pouvez provoquer l'échec de la transaction en émettant une requête, par exemple `g.inject(0).fail('rollback')` avant de fermer le client.

> **Note**
>
> Une requête comme`g.inject(0).fail('rollback')`, utilisée pour générer intentionnellement une erreur afin de forcer une annulation, produit une exception sur le client. Détectez et supprimez l'exception qui en résulte avant de fermer le client.

Pour le bytecode, la transaction peut être contrôlée de manière explicite et la session gérée de manière transparente. Les variantes du langage Gremlin (GLV) prennent en charge la syntaxe `tx()` de Gremlin pour effectuer des opérations de type `commit()` ou `rollback()` d'une transaction, comme suit :

```
GraphTraversalSource g = traversal().withRemote(conn);

Transaction tx = g.tx();

// Spawn a GraphTraversalSource from the Transaction.
// Traversals spawned from gtx are executed within a single transaction.
```

```
GraphTraversalSource gtx = tx.begin();
try {
    gtx.addV('person').iterate();
    gtx.addV('software').iterate();

    tx.commit();
} finally {
    if (tx.isOpen()) {
        tx.rollback();
    }
}
```

Bien que l'exemple ci-dessus soit écrit en Java, vous pouvez également utiliser cette `tx()` syntaxe dans d'autres langages. Pour connaître la syntaxe des transactions spécifique au langage, consultez la section Transactions de la TinkerPop documentation Apache pour Java, Python, Javascript, .NET et Go.

> **Warning**
>
> Les requêtes en lecture seule sans session sont exécutées sous un isolement SNAPSHOT, mais les requêtes en lecture seule exécutées dans le cadre d'une transaction explicite sont exécutées sous un isolement SERIALIZABLE. Les requêtes en lecture seule exécutées sous un isolement `SERIALIZABLE` entraînent une surcharge plus importante et peuvent bloquer les écritures simultanées ou être bloquées par ces dernières, contrairement à celles exécutées sous un isolement `SNAPSHOT`.

Comportement du délai d'expiration pour la validation et l'annulation du bytecode

Lorsque vous utilisez des transactions basées sur un bytecode avec la `tx()` syntaxe, les `rollback()` opérations `commit()` et ne sont pas soumises aux paramètres de délai d'expiration des requêtes. Ni le `neptune_query_timeout` paramètre global ni les valeurs de délai d'expiration par requête définis ne `evaluationTimeout` s'appliquent à ces opérations. Sur le serveur, `commit()` et `rollback()` exécutez-les sans limite de temps jusqu'à ce qu'ils soient terminés ou qu'ils rencontrent une erreur.

Du côté client, les `tx.rollback()` appels `tx.commit()` et les appels du pilote Gremlin ne seront pas terminés tant que le serveur ne répondra pas. Selon la langue, cela peut se manifester sous la forme d'un appel bloquant ou d'une opération asynchrone non résolue. Aucun pilote ne fournit de paramètre de délai d'expiration intégré qui limite ces appels. Consultez la documentation de l'API

correspondant à votre variante de langage G705 spécifique pour plus de détails sur le comportement simultané associé à ces fonctionnalités de transaction.

> **Important**
>
> Si un `rollback()` appel `commit()` ou prend plus de temps que prévu, il est possible qu'il soit bloqué en raison d'un conflit de verrouillage lié à une transaction simultanée. Pour plus d'informations sur les conflits de verrouillage, consultezRésolution des conflits à l'aide de Lock-Wait délais.

Si vous devez limiter le temps d'attente de votre application pour un `commit()` ou`rollback()`, vous pouvez utiliser les fonctionnalités de simultanéité de votre langue pour appliquer un délai d'attente côté client. Si le délai d'attente côté client se déclenche, le serveur poursuit le traitement de l'opération. L'opération côté serveur maintient un thread de travail jusqu'à ce qu'il soit terminé. Après un délai d'attente côté client, fermez la connexion et créez-en une nouvelle plutôt que de réutiliser la connexion existante, car l'état de la transaction est indéterminé.

Server-side nettoyage des transactions

Si un client déconnecte ou abandonne une transaction sans la valider ou l'annuler, Neptune dispose de mécanismes côté serveur qui finissent par nettoyer la transaction orpheline :

- Expiration de session : les Bytecode-based sessions qui restent inactives pendant une durée supérieure à la durée de vie maximale (10 minutes) sont fermées et toute transaction ouverte est annulée.
- Délai d'inactivité des connexions : Neptune WebSocket ferme les connexions inactives depuis environ 20 minutes. Lorsque la connexion se ferme, le serveur annule toute transaction ouverte associée à cette connexion.

Ces mécanismes de nettoyage sont des filets de sécurité. Nous vous recommandons de toujours valider ou annuler explicitement les transactions lorsque vous en avez terminé avec elles.

Diffusion en continu des résultats de requêtes avec Gremlin

Lorsque vous exécutez une traversée Gremlin qui renvoie un grand nombre de résultats, Neptune les renvoie au client par lots via la connexion. WebSocket Neptune envoie les lots de résultats au fur et à mesure qu'ils sont produits, sans attendre que le client en demande davantage. Cela peut être

avantageux si vous souhaitez traiter les résultats tels qu'ils sont renvoyés par le serveur, mais cela nécessite d'utiliser des modèles d'itération différée pour éviter de collecter l'ensemble des résultats dans la mémoire.

Neptune envoie les résultats par lots de 64 par WebSocket image par défaut. Vous ne pouvez pas modifier cette valeur par défaut côté serveur, mais le client peut modifier la taille du lot au cas par cas à l'aide de l'option `batchSize`request (appelée `Tokens.ARGS_BATCH_SIZE` dans le pilote Java ou `connectionPool.resultIterationBatchSize` par défaut au niveau du pilote).

Pour plus de détails sur la configuration des pilotes `batchSize` dans d'autres langues, consultez la section Configuration de chaque pilote dans la documentation sur les pilotes et variantes d'Apache TinkerPop Gremlin.

Comme le serveur envoie automatiquement les résultats, la contre-pression côté client est gérée implicitement via le protocole TCP et le contrôle de flux. WebSocket Si le client met du temps à lire depuis le socket, les écritures du serveur finiront par être bloquées jusqu'à ce que le client rattrape son retard.

> **Important**
>
> Le streaming est particulièrement efficace avec les traversées qui peuvent produire des résultats de manière incrémentielle. Les traversées qui incluent`order()`,`groupCount()`, `group()``dedup()`, ou d'autres étapes nécessitant la réalisation de la traversée complète avant d'émettre des résultats obligeront Neptune à matérialiser l'ensemble de résultats en mémoire avant le début de la diffusion. Dans ces cas, le traitement par lots réduit tout de même la charge de sérialisation par image, mais ne réduit pas l'utilisation de la mémoire côté serveur.

Consommer les résultats de manière incrémentielle

Pour traiter les résultats au fur et à mesure qu'ils arrivent, itérez paresseusement en utilisant`hasNext()`/`next()`ou des API équivalentes plutôt que de rassembler tous les résultats dans une liste. Vous pouvez l'utiliser `next(batchSize)` pour extraire les résultats dans des lots au niveau de l'application, ce qui vous permet d'effectuer un travail intermédiaire entre les lots pendant que le serveur continue de produire des résultats.

Example Java (bytecode GLV)

```
GraphTraversalSource g = traversal().withRemote(connection);

int batchSize = 10;
int batchNum = 0;
var traversal = g.V().hasLabel("movie").values("title").limit(1000);
while (traversal.hasNext()) {
    var batch = traversal.next(batchSize);
    batchNum++;
    for (var title : batch) {
        System.out.println("  " + title);
    }

    // Do other intermediary work here between batch calls
    System.out.println("Batch " + batchNum + " processing complete\n");
}
```

Example Python

```
g = traversal().with_remote(connection)

BATCH_SIZE = 10
batch_num = 0
t = g.V().has_label('movie').values('title').limit(1000)
while t.has_next():
    batch = t.next(BATCH_SIZE)
    batch_num += 1
    for title in batch:
        print(f"  {title}")

    # Do other intermediary work here between batch calls
    print(f"Batch {batch_num} processing complete\n")
```

Example Go

```
// The Go driver does not support next(n), so batches are accumulated manually.
g := gremlingo.Traversal_().WithRemote(connection)

resultSet, err := g.V().HasLabel("movie").Values("title").Limit(1000).GetResultSet()
if err != nil {
    log.Fatal(err)
```

```
}

batchSize := 10
batchNum := 0
for {
    var batch []interface{}
    for i := 0; i < batchSize; i++ {
        result, ok, err := resultSet.One() // returns (value, ok, error); ok is false
 when results are exhausted
        if err != nil {
            log.Fatal(err)
        }
        if !ok {
            break
        }
        batch = append(batch, result)
    }
    if len(batch) == 0 {
        break
    }
    batchNum++
    for _, v := range batch {
        fmt.Printf("  %v\n", v)
    }

    // Do other intermediary work here between batch calls
    fmt.Printf("Batch %d processing complete\n\n", batchNum)
}
```

Example.NET

```
var g = Traversal().WithRemote(connection);

var batchSize = 10;
var batchNum = 0;
var traversal = g.V().HasLabel("movie").Values<string>("title").Limit<string>(1000);
while (traversal.HasNext())
{
    var batch = traversal.Next(batchSize);
    batchNum++;
    foreach (var title in batch)
    {
        Console.WriteLine($"  {title}");
```

```
    }

    // Do other intermediary work here between batch calls
    Console.WriteLine($"Batch {batchNum} processing complete\n");
}
```

Example Node.js

```
// The Node.js driver does not support next(n), so batches are accumulated manually.
const g = traversal().withRemote(connection);

const batchSize = 10;
let batchNum = 0;
const t = g.V().hasLabel('movie').values('title').limit(1000);
while (true) {
    const batch = [];
    for (let i = 0; i < batchSize; i++) {
        const result = await t.next();
        if (result.done) break;
        batch.push(result.value);
    }
    if (batch.length === 0) break;
    batchNum++;
    for (const title of batch) {
        console.log(`  ${title}`);
    }

    // Do other intermediary work here between batch calls
    console.log(`Batch ${batchNum} processing complete\n`);
}
```

Consommation rapide ou consommation incrémentielle

Le streaming vous permet de traiter les résultats de manière incrémentielle au fur et à mesure que des données supplémentaires sont récupérées et renvoyées. Les méthodes suivantes bloquent jusqu'à ce que la totalité du jeu de résultats soit collectée en mémoire, empêchant ainsi votre application d'agir sur les résultats lorsqu'ils arrivent :

- Java : `toList()` ou `toSet()`
- Python : `toList()` ou `toSet()`
- Allez : `ToList()ToSet()`, ou `GetResultSet().GetAll()`

- .NET : `ToList()` ou `Promise()`
- Node.js: `toList()`

> **Note**
>
> Les données circulent toujours de manière incrémentielle sur la WebSocket connexion, même en utilisant ces méthodes. La différence est que votre demande ne peut pas traiter les résultats individuels tant que la collection complète n'est pas complète. Pour traiter les résultats au fur et à mesure qu'ils arrivent, utilisez les modèles d'itération différée ou de traitement par lots illustrés dans les exemples ci-dessus.

Utilisation de l'API Gremlin avec Amazon Neptune

> **Note**
>
> Amazon Neptune ne prend pas en charge la propriété `bindings`.

Les demandes HTTPS Gremlin utilisent un point de terminaison unique : `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin`. Toutes les connexions Neptune doivent utiliser HTTPS.

Vous pouvez connecter la console G705 à un graphe Neptune directement via. WebSockets

Pour plus d'informations sur la connexion au point de terminaison Gremlin, consultez Accès au graphe Neptune avec Gremlin.

L'implémentation Amazon Neptune de Gremlin implique des détails et des différences spécifiques que vous devez prendre en compte. Pour de plus amples informations, veuillez consulter Conformité d'Amazon Neptune avec les normes Gremlin.

Pour plus d'informations sur le langage Gremlin et les traversales, consultez The Traversal dans la documentation d'Apache. TinkerPop

Mise en cache des résultats de requête dans Amazon Neptune Gremlin

Amazon Neptune prend en charge un cache de résultats pour les requêtes Gremlin.

Vous pouvez activer ce cache, puis utiliser un indicateur de requête pour mettre en cache les résultats d'une requête Gremlin en lecture seule.

Toute nouvelle exécution de la requête permet ensuite de récupérer les résultats mis en cache avec une faible latence et sans I/O frais, tant qu'ils sont toujours dans le cache. Ce comportement s'applique pour les requêtes soumises à la fois sur un point de terminaison HTTP et à l'aide de Websockets, sous forme de bytecode ou de chaîne.

> **Note**
>
> Les requêtes envoyées au point de terminaison de profil ne sont pas mises en cache même lorsque le cache de requêtes est activé.

Vous pouvez contrôler le comportement du cache des résultats de requête Neptune de plusieurs manières. Par exemple :

- Vous pouvez obtenir les résultats mis en cache paginés, par blocs.
- Vous pouvez spécifier la durée de vie (TTL) de requêtes spécifiques.
- Vous pouvez vider le cache pour certaines requêtes.
- Vous pouvez vider tout le cache.
- Vous pouvez configurer le cache afin d'être averti si les résultats dépassent sa taille.

Le cache est géré selon une politique LRU (Least-Recently-Used). Par conséquent, une fois que l'espace alloué au cache est plein, les résultats les moins récemment utilisés sont supprimés pour laisser de la place aux nouveaux résultats mis en cache.

> **Important**
>
> Le cache des résultats de requête n'est pas disponible sur les types d'instance `t3.medium` et `t4.medium`.

Activation du cache des résultats de requête dans Neptune

Le cache des résultats des requêtes peut être activé sur toutes les instances d'un cluster ou par instance. Pour activer le cache des résultats sur toutes les instances d'un cluster, définissez le

`neptune_result_cache` paramètre dans le cluster `cluster-parameter-group` sur1. Pour l'activer sur une instance spécifique, définissez le `neptune_result_cache` paramètre de l'instance `instance-parameter-group` sur1. Le paramètre du groupe de paramètres du cluster remplacera la valeur du groupe de paramètres d'instance.

Un redémarrage est nécessaire sur toutes les instances concernées pour que les paramètres du cache de résultats soient appliqués. Bien que vous puissiez activer le cache des résultats sur toutes les instances d'un cluster via le`cluster-parameter-group`, chaque instance conserve son propre cache. La fonctionnalité de cache des résultats de requête n'est pas un cache à l'échelle du cluster.

Une fois le cache des résultats activé, Neptune réserve une partie de la mémoire actuelle à la mise en cache des résultats de requête. Plus le type d'instance que vous utilisez est grand et plus la quantité de mémoire disponible est importante, plus Neptune réserve de mémoire au cache.

Si le cache de mémoire des résultats arrive à saturation, Neptune supprime automatiquement les résultats mis en cache les moins récemment utilisés (LRU) pour laisser de la place à de nouveaux résultats.

Vous pouvez vérifier le statut actuel du cache de résultats à l'aide de la commande Statut d'une instance.

Utilisation d'indicateurs pour mettre en cache les résultats de requête

Une fois le cache des résultats de requête activé, vous pouvez utiliser des indicateurs de requête pour contrôler la mise en cache des requêtes. Tous les exemples ci-dessous s'appliquent à la même traversée de requêtes, à savoir :

```
g.V().has('genre','drama').in('likes')
```

En utilisant **`enable ResultCache`**

Lorsque le cache des résultats de requête est activé, vous pouvez mettre en cache les résultats d'une requête Gremlin à l'aide de l'indicateur de requête `enableResultCache`, comme suit :

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Neptune vous renvoie ensuite les résultats de la requête et les met également en cache. Plus tard, vous pourrez accéder aux résultats mis en cache en émettant à nouveau exactement la même requête :

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

La clé de cache qui identifie les résultats mis en cache est la chaîne de requête elle-même, à savoir :

```
g.V().has('genre','drama').in('likes')
```

En utilisant **`enable ResultCacheWithTTL`**

Vous pouvez spécifier la durée pendant laquelle les résultats de la requête doivent être mis en cache à l'aide de l'indicateur de requête `enableResultCacheWithTTL`. Par exemple, la requête suivante indique que les résultats de la requête doivent expirer au bout de 120 secondes :

```
g.with('Neptune#enableResultCacheWithTTL', 120)
 .V().has('genre','drama').in('likes')
```

Là aussi, la clé de cache qui identifie les résultats mis en cache est la chaîne de requête de base :

```
g.V().has('genre','drama').in('likes')
```

Et là aussi, vous pouvez accéder aux résultats mis en cache à l'aide de cette chaîne de requête avec l'indicateur de requête `enableResultCache` :

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Si 120 secondes ou plus se sont écoulées depuis la mise en cache des résultats, cette requête renvoie de nouveaux résultats et les met en cache, sans aucune durée de vie.

Vous pouvez également accéder aux résultats mis en cache en émettant à nouveau la même requête avec l'indicateur de requête `enableResultCacheWithTTL`. Par exemple :

```
g.with('Neptune#enableResultCacheWithTTL', 140)
 .V().has('genre','drama').in('likes')
```

Tant que 120 secondes ne se sont pas écoulées (ce qui correspond au TTL actuellement en vigueur), cette nouvelle requête avec l'indicateur `enableResultCacheWithTTL` renvoie les résultats mis en cache. Au bout de 120 secondes, elle renvoie de nouveaux résultats et les met en cache avec une durée de vie de 140 secondes.

> **Note**
>
> Si les résultats d'une clé de requête sont déjà mis en cache, la même clé de requête avec `enableResultCacheWithTTL` ne génère pas de nouveaux résultats et n'a aucun effet sur la durée de vie des résultats actuellement mis en cache.
>
> - Si les résultats ont déjà été mis en cache avec `enableResultCache`, le cache doit d'abord être explicitement vidé avant qu'`enableResultCacheWithTTL` génère de nouveaux résultats et les mette en cache pour le TTL qu'il spécifie.
> - Si les résultats ont déjà été mis en cache à l'aide de l'indicateur `enableResultCachewithTTL`, cette durée de vie précédente, ou TTL, doit d'abord expirer avant qu'`enableResultCacheWithTTL` génère de nouveaux résultats et les mette en cache pour la durée de vie qu'il spécifie.

Utilisation de l'**`invalidation ResultCacheKey`**

Vous pouvez utiliser l'indicateur de requête `invalidateResultCacheKey` pour effacer les résultats mis en cache pour une requête particulière. Par exemple :

```
g.with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Cette requête vide la partie du cache correspondant à la clé de requête, `g.V().has('genre','drama').in('likes')`, et renvoie de nouveaux résultats pour cette requête.

Vous pouvez également combiner `invalidateResultCacheKey` avec `enableResultCache` ou `enableResultCacheWithTTL`. Par exemple, la requête suivante efface les résultats actuellement mis en cache, met en cache les nouveaux résultats et les renvoie :

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Utilisation de l'**`invalidation ResultCache`**

Vous pouvez utiliser l'indicateur de requête `invalidateResultCache` pour effacer tous les résultats mis en cache dans le cache des résultats de requête. Par exemple :

```
g.with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Cette requête vide l'intégralité du cache de résultats et renvoie de nouveaux résultats pour la requête.

Vous pouvez également combiner `invalidateResultCache` avec `enableResultCache` ou `enableResultCacheWithTTL`. Par exemple, la requête suivante vide l'intégralité du cache de résultats, met en cache les nouveaux résultats pour cette requête et les renvoie :

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Pagination des résultats de requête mis en cache

Supposons que vous ayez déjà mis en cache un grand nombre de résultats comme suit :

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Supposons maintenant que vous émettiez la requête de plage suivante :

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes').range(0,10)
```

Neptune recherche d'abord la clé de cache complète, à savoir `g.V().has('genre','drama').in('likes').range(0,10)`. Si elle n'existe pas, Neptune vérifie ensuite s'il existe une clé pour cette chaîne de requête sans la plage (à savoir `g.V().has('genre','drama').in('likes')`). Lorsqu'il trouve cette clé, Neptune extrait les dix premiers résultats de son cache, conformément à la plage spécifiée.

> **Note**
>
> Si vous utilisez l'indicateur `invalidateResultCacheKey` avec une requête contenant une plage à la fin, Neptune vide la partie du cache correspondant à une requête sans plage s'il ne trouve pas de correspondance exacte entre cette requête et cette plage.

Utiliser **`num ResultsCached`** avec **`.iterate`** ()

À l'aide de l'indicateur de requête `numResultsCached`, vous pouvez remplir le cache de résultats sans avoir à renvoyer tous les résultats mis en cache, ce qui peut être utile lorsque vous préférez paginer un grand nombre de résultats.

L'indicateur de requête `numResultsCached` ne fonctionne qu'avec les requêtes se terminant par `iterate()`.

Par exemple, si vous souhaitez mettre en cache les 50 premiers résultats de cet exemple de requête :

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').iterate()
```

Dans ce cas, la clé de requête dans le cache est : `g.with("Neptune#numResultsCached", 50).V().has('genre','drama').in('likes')`. Vous pouvez désormais récupérer les 10 premiers résultats mis en cache avec cette requête :

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').range(0, 10)
```

Et vous pouvez récupérer les 10 résultats suivants de la requête comme suit :

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').range(10, 20)
```

N'oubliez pas d'inclure l'indicateur `numResultsCached`. Il s'agit d'un élément essentiel de la clé de requête. Il doit donc être présent pour pouvoir accéder aux résultats mis en cache.

Quelques points à garder à l'esprit lors de l'utilisation de **`num ResultsCached`**

- Le nombre que vous indiquez avec **`numResultsCached`** est appliqué à la fin de la requête. Cela signifie, par exemple, que la requête suivante met réellement en cache les résultats dans la plage `(1000, 1500)` :

  ```
  g.with("Neptune#enableResultCache", true)
  ```

```
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).iterate()
```

- Le nombre que vous indiquez avec **`numResultsCached`** spécifie le nombre maximum de résultats à mettre en cache. Cela signifie, par exemple, que la requête suivante met réellement en cache les résultats dans la plage `(1000, 2000)` :

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 100000)
 .V().range(1000, 2000).iterate()
```

- Les résultats mis en cache par les requêtes se terminant par **`.range().iterate()`** ont leur propre plage. Supposons, par exemple, que vous mettiez en cache les résultats à l'aide d'une requête comme celle-ci :

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).iterate()
```

  Pour récupérer les 100 premiers résultats du cache, vous devriez écrire une requête comme celle-ci :

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).range(0, 100)
```

  Ces cent résultats seraient équivalents aux résultats de la requête de base de la plage `(1000, 1100)`.

Clés de cache de requête utilisées pour localiser les résultats mis en cache

Une fois que les résultats d'une requête ont été mis en cache, les requêtes suivantes avec la même clé de cache de requête récupèrent les résultats du cache au lieu d'en générer de nouveaux. La clé de cache d'une requête est évaluée comme suit :

1. Tous les indicateurs de requête liés au cache sont ignorés, à l'exception de `numResultsCached`.
2. Une dernière étape `iterate()` est ignorée.
3. Le reste de la requête est ordonné en fonction de sa représentation en bytecode.

La chaîne générée est comparée à un index des résultats de requête déjà présents dans le cache afin de déterminer la requête a accédé au cache.

Prenons, par exemple, cette requête :

```
g.withSideEffect('Neptune#typePromotion', false).with("Neptune#enableResultCache",
 true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').iterate()
```

Elle sera stockée en tant que version bytecode de celle-ci :

```
g.withSideEffect('Neptune#typePromotion', false)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes')
```

Exceptions liées au cache de résultats

Si les résultats d'une requête que vous essayez de mettre en cache sont trop importants pour tenir dans la mémoire du cache, même après avoir supprimé tout ce qui était déjà mis en cache, Neptune signale une erreur `QueryLimitExceededException`. Aucun résultat n'est renvoyé, et l'exception génère le message d'erreur suivant :

```
The result size is larger than the allocated cache,
      please refer to results cache best practices for options to rerun the query.
```

Vous pouvez supprimer ce message à l'aide de l'indicateur de requête `noCacheExceptions`, comme suit :

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#noCacheExceptions', true)
 .V().has('genre','drama').in('likes')
```

Réaliser des upserts efficaces avec les étapes Gremlin MergeV () et MergeE ()

Une insertion conditionnelle (également appelée « upsert ») réutilise un sommet ou une arête qui existe déjà, ou crée l'objet nécessaire dans le cas contraire. Des upserts efficaces peuvent faire une différence significative dans les performances des requêtes Gremlin.

Les upserts vous permettent d'écrire des opérations d'insertion idempotentes : quel que soit le nombre de fois que vous exécutez cette opération, le résultat global est le même. Cela est utile dans les scénarios d'écriture hautement simultanés où les modifications simultanées apportées à la même partie du graphe peuvent forcer une ou plusieurs transactions à revenir en arrière avec une exception `ConcurrentModificationException`, nécessitant ainsi de nouvelles tentatives.

> **Note**
>
> Les `mergeE()` étapes `mergeV()` et nécessitent la version 1.2.1.0 ou ultérieure du moteur Neptune, qui a introduit le support de la version 3.6. TinkerPop Si votre cluster exécute une ancienne version du moteur, consultez Upserts efficaces avant la version 3.6.x plutôt. Vous pouvez vérifier la version actuelle de votre moteur à l'aide du point de terminaison d'état de l'instance.

Par exemple, la requête suivante insère un sommet en utilisant l'élément `Map` fourni pour essayer d'abord de trouver un sommet avec le `T.id "v-1"`. Si ce sommet est trouvé, il est renvoyé. Dans le cas contraire, un sommet contenant cet `id` et cette propriété est créé par le biais de la clause `onCreate`.

```
g.mergeV([(id):'v-1']).
  option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org'])
```

Exécution d'upserts par lots pour améliorer le débit

Pour les scénarios d'écriture à haut débit, vous pouvez enchaîner les étapes `mergeV()` et `mergeE()` pour effectuer l'upsert en bloc des sommets et des arêtes. Le traitement par lots réduit la charge transactionnelle liée à l'insertion par upsert d'un grand nombre de sommets et d'arêtes. Vous pouvez ainsi améliorer davantage le débit en augmentant les demandes d'upserts par lots en parallèle à l'aide de plusieurs clients.

En règle générale, nous recommandons d'insérer par upsert environ 200 enregistrements par demande par lots. Un enregistrement correspond à une étiquette ou propriété individuelle de sommet ou d'arête. Par exemple, un sommet doté d'une seule étiquette et de quatre propriétés génère cinq enregistrements. Une arête dotée d'une étiquette et d'une seule propriété génère deux enregistrements. Si vous souhaitez insérer par upsert des lots de sommets, chacun avec une seule étiquette et quatre propriétés, vous devez commencer par une taille de lot de 40, car `200 / (1 + 4) = 40`.

Vous pouvez tester différentes tailles de lots. 200 enregistrements par lot constituent un bon point de départ, mais la taille de lot idéale peut être supérieure ou inférieure en fonction de votre charge de travail. Notez toutefois que Neptune peut limiter le nombre total d'étapes Gremlin par demande. Cette limite n'est pas documentée, mais par mesure de sécurité, essayez de faire en sorte que les demandes ne contiennent pas plus de 1 500 étapes Gremlin. Neptune peut rejeter des demandes en bloc volumineuses comportant plus de 1 500 étapes.

Pour augmenter le débit, vous pouvez insérer par upsert des lots en parallèle à l'aide de plusieurs clients (voir Création d'écritures Gremlin multithreads efficaces). Le nombre de clients doit être identique au nombre de threads de travail de l'instance d'enregistreur Neptune, qui correspond généralement au nombre de vCPU sur le serveur, multiplié par deux. Par exemple, une instance `r5.8xlarge` possède 32 vCPU et 64 threads de travail. Pour les scénarios d'écriture à haut débit utilisant une instance `r5.8xlarge`, vous devez utiliser 64 clients écrivant des upserts par lots sur Neptune en parallèle.

Chaque client doit soumettre une demande par lots et attendre qu'elle soit terminée avant de soumettre une autre demande. Bien que les différents clients fonctionnent en parallèle, chacun d'eux soumet des demandes en série. Cela garantit que le serveur reçoit un flux constant de demandes qui occupent tous les threads de travail sans encombrer la file d'attente des demandes côté serveur (voir Dimensionnement des instances de base de données dans un cluster de bases de données Neptune).

Essayer d'éviter les étapes qui génèrent plusieurs traverseurs

Lorsqu'une étape Gremlin s'exécute, elle utilise un traverseur entrant et émet un ou plusieurs traverseurs de sortie. Le nombre de traverseurs émis par une étape détermine le nombre de fois que l'étape suivante sera exécutée.

Généralement, lorsque vous effectuez des opérations par lots, vous souhaitez que chaque opération, telle que l'upsert du sommet A, soit exécutée une seule fois, de sorte que la séquence des opérations ressemble à ceci : upsert du sommet A, puis upsert du sommet B, puis upsert du sommet C, etc. Tant qu'une étape ne crée ou ne modifie qu'un seul élément, elle n'émet qu'un seul traverseur, et les étapes représentant l'opération suivante ne sont exécutées qu'une seule fois. Si, en revanche, une opération crée ou modifie plusieurs éléments, elle émet plusieurs traverseurs, ce qui entraîne l'exécution des étapes suivantes plusieurs fois, une fois par traverseur émis. Cela peut obliger la base de données à effectuer des tâches supplémentaires inutiles et, dans certains cas, à créer des sommets, des arêtes ou des valeurs de propriétés supplémentaires superflus.

La requête `g.V().addV()` est un bon exemple de cas où la situation peut dégénérer. Cette requête simple ajoute un sommet pour chaque sommet du graphe, car `V()` émet un traverseur pour chaque sommet du graphe et chacun de ces traverseurs déclenche un appel à `addV()`.

Consultez Combinaison d'upserts et d'insertions pour découvrir comment gérer les opérations qui peuvent émettre plusieurs traverseurs.

Insertion de sommets par upsert

L'étape `mergeV()` est spécialement conçue pour l'upsert de sommets. Elle utilise comme argument un objet `Map` qui représente les éléments correspondant aux sommets existants dans le graphe. Si aucun élément n'est trouvé, elle utilise cet objet `Map` pour créer un sommet. Cette étape vous permet également de modifier le comportement en cas de création ou de correspondance. Le modulateur `option()` peut alors être associé à des jetons `Merge.onCreate` et `Merge.onMatch` pour contrôler ces comportements respectifs. Consultez la documentation de TinkerPop référence pour plus d'informations sur l'utilisation de cette étape.

Vous pouvez utiliser un ID de sommet pour déterminer si un sommet spécifique existe. Il s'agit de l'approche préférée, car Neptune optimise les upserts pour les cas d'utilisation hautement simultanés liés aux ID. Par exemple, la requête suivante crée un sommet avec un ID de sommet donné s'il n'existe pas déjà, ou le réutilise s'il existe déjà :

```
g.mergeV([(T.id): 'v-1']).
    option(onCreate, [(T.label): 'PERSON', email: 'person-1@example.org', age: 21]).
    option(onMatch, [age: 22]).
  id()
```

Notez que cette requête se termine par une étape `id()`. Bien que cela ne soit pas strictement nécessaire pour réaliser l'upsert du sommet, une étape `id()` à la fin d'une requête d'upsert garantit que le serveur ne sérialise pas toutes les propriétés du sommet vers le client, ce qui contribue à réduire le coût de verrouillage de la requête.

Vous pouvez également utiliser une propriété de sommet pour identifier un sommet :

```
g.mergeV([email: 'person-1@example.org']).
    option(onCreate, [(T.label): 'PERSON', age: 21]).
    option(onMatch, [age: 22]).
  id()
```

Si possible, utilisez vos propres ID fournis par l'utilisateur pour créer des sommets, et servez-vous de ces ID pour déterminer si un sommet existe lors d'une opération d'upsert. Cela permet à Neptune d'optimiser les upserts. Une ID-based modification peut être nettement plus efficace qu'une modification basée sur des propriétés lorsque des modifications simultanées sont courantes.

Enchaînement d'upserts de sommets

Vous pouvez enchaîner des upserts de sommets pour les insérer dans un lot :

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org'))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org'))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-3')
                         .property('email', 'person-3@example.org'))
 .id()
```

Vous pouvez également utiliser cette syntaxe `mergeV()` :

```
g.mergeV([(T.id): 'v-1', (T.label): 'PERSON', email: 'person-1@example.org']).
  mergeV([(T.id): 'v-2', (T.label): 'PERSON', email: 'person-2@example.org']).
  mergeV([(T.id): 'v-3', (T.label): 'PERSON', email: 'person-3@example.org'])
```

Cependant, comme cette forme de requête inclut des éléments dans les critères de recherche qui sont superflus par rapport à la recherche de base par `id`, elle n'est pas aussi efficace que la requête précédente.

Exécution d'upserts d'arêtes

L'étape `mergeE()` est spécialement conçue pour l'upsert d'arêtes. Elle utilise comme argument un objet `Map` qui représente les éléments correspondant aux arêtes existantes dans le graphe. Si aucun élément n'est trouvé, elle utilise cet objet `Map` pour créer une arête. Cette étape vous permet

également de modifier le comportement en cas de création ou de correspondance. Le modulateur `option()` peut alors être associé à des jetons `Merge.onCreate` et `Merge.onMatch` pour contrôler ces comportements respectifs. Consultez la documentation de TinkerPop référence pour plus d'informations sur l'utilisation de cette étape.

Vous pouvez utiliser les ID d'arêtes pour insérer des arêtes par upsert tout comme vous exécutez des upserts de sommets à l'aide d'ID de sommet personnalisés. Là aussi, il s'agit de l'approche préférée, car elle permet à Neptune d'optimiser la requête. Par exemple, la requête suivante crée une arête en fonction de son ID d'arête si elle n'existe pas déjà, ou la réutilise si elle existe déjà. Cette requête utilise également les ID des sommets `Direction.from` et `Direction.to` si elle doit créer une arête :

```
g.mergeE([(T.id): 'e-1']).
    option(onCreate, [(from): 'v-1', (to): 'v-2', weight: 1.0]).
    option(onMatch, [weight: 0.5]).
  id()
```

Notez que cette requête se termine par une étape `id()`. Bien que cela ne soit pas strictement nécessaire pour réaliser l'upsert de l'arête, une étape `id()` à la fin d'une requête d'upsert garantit que le serveur ne sérialise pas toutes les propriétés de l'arête vers le client, ce qui contribue à réduire le coût de verrouillage de la requête.

De nombreuses applications utilisent des ID de sommet personnalisés, mais laissent à Neptune le soin de générer les ID d'arête. Si vous ne connaissez pas l'ID d'une arête, mais que vous connaissez les ID de sommet `from` et `to`, vous pouvez utiliser ce type de requête pour réaliser l'upsert d'une arête :

```
g.mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  id()
```

Tous les sommets référencés par `mergeE()` doivent exister pour que l'étape crée l'arête.

Enchaînement d'upserts d'arêtes

Comme pour les upserts de sommets, il est simple d'enchaîner les étapes `mergeE()` pour les demandes en bloc :

```
g.mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  mergeE([(from): 'v-2', (to): 'v-3', (T.label): 'KNOWS']).
  mergeE([(from): 'v-3', (to): 'v-4', (T.label): 'KNOWS']).
```

```
  id()
```

Combinaison d'upserts de sommets et d'arêtes

Parfois, il peut être utile d'insérer par upsert à la fois les sommets et les arêtes qui les relient. Vous pouvez combiner les exemples de lots présentés ici. L'exemple suivant insère par upsert trois sommets et deux arêtes :

```
g.mergeV([(id):'v-1']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org']).
  mergeV([(id):'v-2']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-2@example.org']).
  mergeV([(id):'v-3']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-3@example.org']).
  mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  mergeE([(from): 'v-2', (to): 'v-3', (T.label): 'KNOWS']).
 id()
```

Combinaison d'upserts et d'insertions

Parfois, il peut être utile d'insérer par upsert à la fois les sommets et les arêtes qui les relient. Vous pouvez combiner les exemples de lots présentés ici. L'exemple suivant insère par upsert trois sommets et deux arêtes :

Les upserts traitent généralement un élément à la fois. Si vous vous en tenez aux modèles d'upsert présentés ici, chaque opération d'upsert émet un seul traverseur, ce qui entraîne l'exécution de l'opération suivante une seule fois.

Cependant, il peut arriver que vous souhaitiez combiner des upserts avec des insertions. Cela peut notamment être le cas si vous utilisez des arêtes pour représenter des instances d'actions ou d'événements. Une demande peut utiliser des upserts pour s'assurer que tous les sommets nécessaires existent, puis utiliser des insertions pour ajouter des arêtes. Avec les demandes de ce type, soyez attentif au nombre potentiel de traverseurs émis par chaque opération.

Prenons l'exemple suivant, qui combine des upserts et des insertions pour ajouter des arêtes représentant des événements dans le graphe :

```
// Fully optimized, but inserts too many edges
g.mergeV([(id):'v-1']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org']).
  mergeV([(id):'v-2']).
```

```
    option(onCreate, [(label): 'PERSON', 'email': 'person-2@example.org']).
  mergeV([(id):'v-3']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-3@example.org']).
  mergeV([(T.id): 'c-1', (T.label): 'CITY', name: 'city-1']).
  V('p-1', 'p-2').
  addE('FOLLOWED').to(V('p-1')).
  V('p-1', 'p-2', 'p-3').
  addE('VISITED').to(V('c-1')).
  id()
```

La requête doit insérer cinq arêtes : deux arêtes SUIVIES et trois arêtes VISITÉES. Cependant, la requête telle qu'elle est écrite insère huit arêtes : deux arêtes SUIVIES et six arêtes VISITÉES. Cela est dû au fait que l'opération qui insère les deux arêtes suivies émet deux traverseurs, ce qui entraîne l'exécution de l'opération suivante d'insertion de trois arêtes deux fois.

La solution consiste à ajouter une étape `fold()` après chaque opération susceptible d'émettre plusieurs traverseurs :

```
g.mergeV([(T.id): 'v-1', (T.label): 'PERSON', email: 'person-1@example.org']).
  mergeV([(T.id): 'v-2', (T.label): 'PERSON', email: 'person-2@example.org']).
  mergeV([(T.id): 'v-3', (T.label): 'PERSON', email: 'person-3@example.org']).
  mergeV([(T.id): 'c-1', (T.label): 'CITY', name: 'city-1']).
  V('p-1', 'p-2').
  addE('FOLLOWED').
    to(V('p-1')).
  fold().
  V('p-1', 'p-2', 'p-3').
  addE('VISITED').
    to(V('c-1')).
  id()
```

Nous avons inséré ici une étape `fold()` après l'opération qui insère les arêtes SUIVIES. Il en résulte un seul traverseur, et l'opération suivante n'est donc exécutée qu'une seule fois.

L'inconvénient de cette approche est que la requête n'est plus entièrement optimisée, car `fold()` n'est pas optimisé. L'opération d'insertion qui suit `fold()` ne sera maintenant pas optimisée non plus.

Si vous devez utiliser `fold()` pour réduire le nombre de traverseurs lors des étapes suivantes, essayez d'organiser les opérations de manière à ce que les moins coûteuses occupent la partie non optimisée de la requête.

Configuration de la cardinalité

La cardinalité par défaut pour les propriétés des sommets dans Neptune est définie, ce qui signifie que lors de l'utilisation de mergeV (), les valeurs fournies sur la carte recevront toutes cette cardinalité. Pour utiliser la cardinalité unique, vous devez être explicite dans son utilisation. À partir de la TinkerPop version 3.7.0, une nouvelle syntaxe permet de fournir la cardinalité dans le cadre de la carte, comme indiqué dans l'exemple suivant :

```
g.mergeV([(T.id): '1234']).
  option(onMatch, ['age': single(20), 'name': single('alice'), 'city': set('miami')])
```

Vous pouvez également définir la cardinalité par défaut pour cela `option` comme suit :

```
// age and name are set to single cardinality by default
g.mergeV([(T.id): '1234']).
  option(onMatch, ['age': 22, 'name': 'alice', 'city': set('boston')], single)
```

Il existe moins d'options pour définir la cardinalité dans les versions `mergeV()` antérieures à 3.7.0. L'approche générale consiste à revenir à l'`property()`étape suivante :

```
g.mergeV([(T.id): '1234']).
  option(onMatch, sideEffect(property(single,'age', 20).
  property(set,'city','miami')).constant([:]))
```

> **Note**
>
> Cette approche ne fonctionnera que `mergeV()` si elle est utilisée avec une étape de démarrage. Vous ne pourriez donc pas `mergeV()` enchaîner un seul parcours, car la première étape utilisant cette syntaxe `mergeV()` après le début produira une erreur si le traverseur entrant est un élément graphique. Dans ce cas, vous souhaiterez diviser vos `mergeV()` appels en plusieurs demandes, chacune pouvant constituer une étape de départ.

Réaliser des upserts Gmail efficaces avec **`fold () /coalesce () / unfold`** ()

Une insertion conditionnelle (également appelée « upsert ») réutilise un sommet ou une arête qui existe déjà, ou crée l'objet nécessaire dans le cas contraire. Des upserts efficaces peuvent faire une différence significative dans les performances des requêtes Gremlin.

Cette page montre comment utiliser le modèle Gremlin `fold()/coalesce()/unfold()` pour réaliser des upserts efficaces. Cependant, avec la sortie de la TinkerPop version 3.6.x introduite dans Neptune dans la version 1.2.1.0 du moteur, les nouvelles `mergeE()` étapes sont préférables dans la plupart `mergeV()` des cas. Le modèle `fold()/coalesce()/unfold()` décrit ici peut encore être utile dans certaines situations complexes, mais en régle générale, utilisez `mergeV()` et `mergeE()` si vous le pouvez, comme décrit dans Réaliser des upserts efficaces avec les étapes Gremlin MergeV () et MergeE ().

Les upserts vous permettent d'écrire des opérations d'insertion idempotentes : quel que soit le nombre de fois que vous exécutez cette opération, le résultat global est le même. Cela est utile dans les scénarios d'écriture hautement simultanés où les modifications simultanées apportées à la même partie du graphe peuvent forcer une ou plusieurs transactions à revenir en arrière avec une exception `ConcurrentModificationException`, nécessitant ainsi une nouvelle tentative.

Par exemple, la requête suivante insère un sommet en recherchant d'abord le sommet spécifié dans le jeu de données, puis en regroupant les résultats dans une liste. Dans la première traversée fournie à l'étape `coalesce()`, la requête déplie ensuite cette liste. Si la liste dépliée n'est pas vide, les résultats sont émis à partir de `coalesce()`. Toutefois, si `unfold()` renvoie une collection vide parce que le sommet n'existe pas, `coalesce()` passe à l'évaluation de la deuxième traversée avec laquelle il a été fourni, et dans cette deuxième traversée, la requête crée le sommet manquant.

```
g.V('v-1').fold()
          .coalesce(
             unfold(),
             addV('Person').property(id, 'v-1')
                           .property('email', 'person-1@example.org')
           )
```

Utiliser une forme optimisée de **`coalesce ()`** pour les upserts

Neptune peut optimiser l'idiome `fold().coalesce(unfold(), ...)` pour effectuer des mises à jour à haut débit, mais cette optimisation ne fonctionne que si les deux parties de l'idiome

`coalesce()` renvoient un sommet ou une arête, mais rien d'autre. Si vous essayez de renvoyer quelque chose de différent, tel qu'une propriété, à partir de n'importe quelle partie du `coalesce()`, l'optimisation Neptune n'a pas lieu. La requête peut aboutir, mais elle ne fonctionnera pas aussi bien qu'une version optimisée, en particulier pour les vastes jeux de données.

Étant donné que les requêtes d'upsert non optimisées augmentent les temps d'exécution et réduisent le débit, il est utile d'utiliser le point de terminaison Gremlin `explain` pour déterminer si une requête d'upsert est entièrement optimisée. Lorsque vous examinez les plans `explain`, recherchez les lignes commençant par `+ not converted into Neptune steps` et `WARNING: >>`. Par exemple :

```
+ not converted into Neptune steps: [FoldStep, CoalesceStep([[UnfoldStep],
 [AddEdgeSte...
WARNING: >> FoldStep << is not supported natively yet
```

Ces avertissements peuvent vous aider à identifier les parties d'une requête qui empêchent son optimisation complète.

Parfois, il n'est pas possible d'optimiser complètement une requête. Dans ces situations, vous devez essayer de placer les étapes qui ne peuvent pas être optimisées à la fin de la requête, afin de permettre au moteur d'optimiser autant d'étapes que possible. Cette technique est utilisée dans certains exemples d'upserts en bloc, où tous les upserts optimisés pour un ensemble de sommets ou d'arêtes sont effectués avant que des modifications supplémentaires potentiellement non optimisées ne soient appliquées aux mêmes sommets ou arêtes.

Exécution d'upserts par lots pour améliorer le débit

Pour les scénarios d'écriture à haut débit, vous pouvez enchaîner les étapes d'upsert pour effectuer l'upsert en bloc des sommets et des arêtes. Le traitement par lots réduit la charge transactionnelle liée à l'insertion par upsert d'un grand nombre de sommets et d'arêtes. Vous pouvez ainsi améliorer davantage le débit en augmentant les demandes d'upserts par lots en parallèle à l'aide de plusieurs clients.

En règle générale, nous recommandons d'insérer par upsert environ 200 enregistrements par demande par lots. Un enregistrement correspond à une étiquette ou propriété individuelle de sommet ou d'arête. Par exemple, un sommet doté d'une seule étiquette et de quatre propriétés génère cinq enregistrements. Une arête dotée d'une étiquette et d'une seule propriété génère deux enregistrements. Si vous souhaitez insérer par upsert des lots de sommets, chacun avec une seule étiquette et quatre propriétés, vous devez commencer par une taille de lot de 40, car `200 / (1 + 4) = 40`.

Vous pouvez tester différentes tailles de lots. 200 enregistrements par lot constituent un bon point de départ, mais la taille de lot idéale peut être supérieure ou inférieure en fonction de votre charge de travail. Notez toutefois que Neptune peut limiter le nombre total d'étapes Gremlin par demande. Cette limite n'est pas documentée, mais par mesure de sécurité, essayez de faire en sorte que les demandes ne contiennent pas plus de 1 500 étapes Gremlin. Neptune peut rejeter des demandes en bloc volumineuses comportant plus de 1 500 étapes.

Pour augmenter le débit, vous pouvez insérer par upsert des lots en parallèle à l'aide de plusieurs clients (voir Création d'écritures Gremlin multithreads efficaces). Le nombre de clients doit être identique au nombre de threads de travail de l'instance d'enregistreur Neptune, qui correspond généralement au nombre de vCPU sur le serveur, multiplié par deux. Par exemple, une instance `r5.8xlarge` possède 32 vCPU et 64 threads de travail. Pour les scénarios d'écriture à haut débit utilisant une instance `r5.8xlarge`, vous devez utiliser 64 clients écrivant des upserts par lots sur Neptune en parallèle.

Chaque client doit soumettre une demande par lots et attendre qu'elle soit terminée avant de soumettre une autre demande. Bien que les différents clients fonctionnent en parallèle, chacun d'eux soumet des demandes en série. Cela garantit que le serveur reçoit un flux constant de demandes qui occupent tous les threads de travail sans encombrer la file d'attente des demandes côté serveur (voir Dimensionnement des instances de base de données dans un cluster de bases de données Neptune).

Essayer d'éviter les étapes qui génèrent plusieurs traverseurs

Lorsqu'une étape Gremlin s'exécute, elle utilise un traverseur entrant et émet un ou plusieurs traverseurs de sortie. Le nombre de traverseurs émis par une étape détermine le nombre de fois que l'étape suivante sera exécutée.

Généralement, lorsque vous effectuez des opérations par lots, vous souhaitez que chaque opération, telle que l'upsert du sommet A, soit exécutée une seule fois, de sorte que la séquence des opérations ressemble à ceci : upsert du sommet A, puis upsert du sommet B, puis upsert du sommet C, etc. Tant qu'une étape ne crée ou ne modifie qu'un seul élément, elle n'émet qu'un seul traverseur, et les étapes représentant l'opération suivante ne sont exécutées qu'une seule fois. Si, en revanche, une opération crée ou modifie plusieurs éléments, elle émet plusieurs traverseurs, ce qui entraîne l'exécution des étapes suivantes plusieurs fois, une fois par traverseur émis. Cela peut obliger la base de données à effectuer des tâches supplémentaires inutiles et, dans certains cas, à créer des sommets, des arêtes ou des valeurs de propriétés supplémentaires superflus.

La requête `g.V().addV()` est un bon exemple de cas où la situation peut dégénérer. Cette requête simple ajoute un sommet pour chaque sommet du graphe, car `V()` émet un traverseur pour chaque sommet du graphe et chacun de ces traverseurs déclenche un appel à `addV()`.

Consultez Combinaison d'upserts et d'insertions pour découvrir comment gérer les opérations qui peuvent émettre plusieurs traverseurs.

Insertion de sommets par upsert

Vous pouvez utiliser un ID de sommet pour déterminer s'il existe un sommet correspondant. Il s'agit de l'approche préférée, car Neptune optimise les upserts pour les cas d'utilisation hautement simultanés liés aux ID. Par exemple, la requête suivante crée un sommet avec un ID de sommet donné s'il n'existe pas déjà, ou le réutilise s'il existe déjà :

```
g.V('v-1')
 .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-1')
                          .property('email', 'person-1@example.org'))
  .id()
```

Notez que cette requête se termine par une étape `id()`. Bien que cela ne soit pas strictement nécessaire pour réaliser l'upsert du sommet, l'ajout d'une étape `id()` à la fin d'une requête d'upsert garantit que le serveur ne sérialise pas toutes les propriétés du sommet vers le client, ce qui contribue à réduire le coût de verrouillage de la requête.

Vous pouvez également utiliser une propriété de sommet pour déterminer si le sommet existe :

```
g.V()
 .hasLabel('Person')
 .has('email', 'person-1@example.org')
 .fold()
 .coalesce(unfold(),
           addV('Person').property('email', 'person-1@example.org'))
 .id()
```

Si possible, utilisez vos propres ID fournis par l'utilisateur pour créer des sommets, et servez-vous de ces ID pour déterminer si un sommet existe lors d'une opération d'upsert. Cela permet à Neptune d'optimiser les upserts par rapport aux ID. Un ID-based bouleversement peut être nettement plus

efficace qu'un bouleversement basé sur des propriétés dans des scénarios de modification très simultanés.

Enchaînement d'upserts de sommets

Vous pouvez enchaîner des upserts de sommets pour les insérer dans un lot :

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org'))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org'))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-3')
                         .property('email', 'person-3@example.org'))
 .id()
```

Exécution d'upserts d'arêtes

Vous pouvez utiliser les ID d'arêtes pour insérer des arêtes par upsert tout comme vous exécutez des upserts de sommets à l'aide d'ID de sommet personnalisés. Là aussi, il s'agit de l'approche préférée, car elle permet à Neptune d'optimiser la requête. Par exemple, la requête suivante crée une arête en fonction de son ID d'arête si elle n'existe pas déjà, ou la réutilise si elle existe déjà. Cette requête utilise également les ID des sommets `from` et `to` si elle doit créer une arête.

```
g.E('e-1')
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2'))
                        .property(id, 'e-1'))
 .id()
```

De nombreuses applications utilisent des ID de sommet personnalisés, mais laissent à Neptune le soin de générer les ID d'arête. Si vous ne connaissez pas l'ID d'une arête, mais que vous connaissez

les ID de sommet `from` et `to`, vous pouvez utiliser cette formulation pour réaliser l'upsert d'une arête :

```
g.V('v-1')
 .outE('KNOWS')
 .where(inV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2')))
 .id()
```

Notez que l'étape du sommet dans la clause `where()` doit être `inV()` (ou `outV()` si vous avez utilisé `inE()` pour trouver l'arête), non pas `otherV()`. N'utilisez pas `otherV()` ici. Dans le cas contraire, la requête ne sera pas optimisée, et les performances en pâtiront. Par exemple, Neptune n'optimiserait pas la requête suivante :

```
// Unoptimized upsert, because of otherV()
g.V('v-1')
 .outE('KNOWS')
 .where(otherV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2')))
 .id()
```

Si vous ne connaissez pas les ID des arêtes ou des sommets dès le départ, vous pouvez les insérer par upsert à l'aide des propriétés des sommets :

```
g.V()
 .hasLabel('Person')
 .has('name', 'person-1')
 .outE('LIVES_IN')
 .where(inV().hasLabel('City').has('name', 'city-1'))
 .fold()
 .coalesce(unfold(),
           addE('LIVES_IN').from(V().hasLabel('Person')
                                    .has('name', 'person-1'))
                           .to(V().hasLabel('City')
                                  .has('name', 'city-1')))
```

```
  .id()
```

Comme pour les upserts, il est préférable d'utiliser des upserts de ID-based bord utilisant un identifiant de bord ou `from` des identifiants de `to` sommet, plutôt que des upserts basés sur des propriétés, afin que Neptune puisse optimiser pleinement l'upserts.

Vérification **`de l'existence des sommets depuis`** et **`vers`**

Notez la structure des étapes qui créent une arête : `addE().from().to()`. Cette structure garantit que la requête vérifie l'existence à la fois du sommet `from` et du sommet `to`. S'ils n'existent pas, la requête renvoie le message d'erreur suivant :

```
{
  "detailedMessage": "Encountered a traverser that does not map to a value for child...
  "code": "IllegalArgumentException",
  "requestId": "..."
}
```

S'il est possible que le sommet `from` ou `to` n'existe pas, vous devez essayer de les placer par upsert avant de réaliser l'upsert de l'arête qui les sépare. Consultez Combinaison d'upserts de sommets et d'arêtes.

Une autre structure permet de créer une arête que vous ne devez pas utiliser : `V().addE().to()`. Elle ajoute une arête uniquement si le sommet `from` existe. Si le sommet `to` n'existe pas, la requête génère une erreur, comme décrit précédemment, mais si le sommet `from` n'existe pas, l'upsert de l'arête échoue en arrière-plan, sans générer d'erreur. Par exemple, si le sommet `from` n'existe pas, l'upsert suivant a lieu sans effectuer l'upsert d'une arête :

```
// Will not insert edge if from vertex does not exist
g.V('v-1')
 .outE('KNOWS')
 .where(inV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
           V('v-1').addE('KNOWS')
                   .to(V('v-2')))
 .id()
```

Enchaînement d'upserts d'arêtes

Si vous souhaitez enchaîner des upserts afin de créer une demande en bloc, vous devez commencer chaque upsert par une recherche de sommet, même si vous connaissez déjà l'ID des arêtes.

Si vous connaissez déjà l'ID des arêtes dont vous souhaitez effectuer l'upsert, ainsi que les ID des sommets `from` et `to`, vous pouvez utiliser cette formulation :

```
g.V('v-1')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
           V('v-1').addE('KNOWS')
                   .to(V('v-2'))
                   .property(id, 'e-1'))
 .V('v-3')
 .outE('KNOWS')
 .hasId('e-2').fold()
 .coalesce(unfold(),
           V('v-3').addE('KNOWS')
                   .to(V('v-4'))
                   .property(id, 'e-2'))
 .V('v-5')
 .outE('KNOWS')
 .hasId('e-3')
 .fold()
 .coalesce(unfold(),
           V('v-5').addE('KNOWS')
                   .to(V('v-6'))
                   .property(id, 'e-3'))
 .id()
```

Connaître les ID de sommet `from` et `to`, mais pas l'ID des arêtes dont vous souhaitez effectuer l'upsert est peut-être le scénario d'upsert des arêtes en bloc le plus courant. Dans ce cas, utilisez la formulation suivante :

```
g.V('v-1')
 .outE('KNOWS')
 .where(inV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
```

```
          V('v-1').addE('KNOWS')
                  .to(V('v-2')))

 .V('v-3')
 .outE('KNOWS')
 .where(inV().hasId('v-4'))
 .fold()
 .coalesce(unfold(),
          V('v-3').addE('KNOWS')
                  .to(V('v-4')))
 .V('v-5')
 .outE('KNOWS')
 .where(inV().hasId('v-6'))
 .fold()
 .coalesce(unfold(),
          V('v-5').addE('KNOWS').to(V('v-6')))
 .id()
```

Si vous connaissez l'ID des arêtes dont vous souhaitez effectuer l'upsert, mais pas les ID des sommets `from` et `to` (ce qui est rare), vous pouvez utiliser cette formulation :

```
g.V()
 .hasLabel('Person')
 .has('email', 'person-1@example.org')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
          V().hasLabel('Person')
             .has('email', 'person-1@example.org')
             .addE('KNOWS')
             .to(V().hasLabel('Person')
                    .has('email', 'person-2@example.org'))
                    .property(id, 'e-1'))
 .V()
 .hasLabel('Person')
 .has('email', 'person-3@example.org')
 .outE('KNOWS')
 .hasId('e-2')
 .fold()
 .coalesce(unfold(),
          V().hasLabel('Person')
             .has('email', 'person-3@example.org')
```

```
              .addE('KNOWS')
              .to(V().hasLabel('Person')
                     .has('email', 'person-4@example.org'))
              .property(id, 'e-2'))
 .V()
 .hasLabel('Person')
 .has('email', 'person-5@example.org')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
           V().hasLabel('Person')
              .has('email', 'person-5@example.org')
              .addE('KNOWS')
              .to(V().hasLabel('Person')
                     .has('email', 'person-6@example.org'))
                     .property(id, 'e-3'))
 .id()
```

Combinaison d'upserts de sommets et d'arêtes

Parfois, il peut être utile d'insérer par upsert à la fois les sommets et les arêtes qui les relient. Vous pouvez combiner les exemples de lots présentés ici. L'exemple suivant insère par upsert trois sommets et deux arêtes :

```
g.V('p-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-1')
                         .property('email', 'person-1@example.org'))
 .V('p-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-2')
                         .property('name', 'person-2@example.org'))
 .V('c-1')
 .fold()
 .coalesce(unfold(),
           addV('City').property(id, 'c-1')
                       .property('name', 'city-1'))
 .V('p-1')
 .outE('LIVES_IN')
 .where(inV().hasId('c-1'))
```

```
  .fold()
  .coalesce(unfold(),
            V('p-1').addE('LIVES_IN')
                    .to(V('c-1')))
  .V('p-2')
  .outE('LIVES_IN')
  .where(inV().hasId('c-1'))
  .fold()
  .coalesce(unfold(),
            V('p-2').addE('LIVES_IN')
                    .to(V('c-1')))
  .id()
```

Combinaison d'upserts et d'insertions

Parfois, il peut être utile d'insérer par upsert à la fois les sommets et les arêtes qui les relient. Vous pouvez combiner les exemples de lots présentés ici. L'exemple suivant insère par upsert trois sommets et deux arêtes :

Les upserts traitent généralement un élément à la fois. Si vous vous en tenez aux modèles d'upsert présentés ici, chaque opération d'upsert émet un seul traverseur, ce qui entraîne l'exécution de l'opération suivante une seule fois.

Cependant, il peut arriver que vous souhaitiez combiner des upserts avec des insertions. Cela peut notamment être le cas si vous utilisez des arêtes pour représenter des instances d'actions ou d'événements. Une demande peut utiliser des upserts pour s'assurer que tous les sommets nécessaires existent, puis utiliser des insertions pour ajouter des arêtes. Avec les demandes de ce type, soyez attentif au nombre potentiel de traverseurs émis par chaque opération.

Prenons l'exemple suivant, qui combine des upserts et des insertions pour ajouter des arêtes représentant des événements dans le graphe :

```
// Fully optimized, but inserts too many edges
g.V('p-1')
  .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'p-1')
                          .property('email', 'person-1@example.org'))
  .V('p-2')
  .fold()
```

```
  .coalesce(unfold(),
            addV('Person').property(id, 'p-2')
                          .property('name', 'person-2@example.org'))
  .V('p-3')
  .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'p-3')
                          .property('name', 'person-3@example.org'))
  .V('c-1')
  .fold()
  .coalesce(unfold(),
            addV('City').property(id, 'c-1')
                        .property('name', 'city-1'))
  .V('p-1', 'p-2')
  .addE('FOLLOWED')
  .to(V('p-1'))
  .V('p-1', 'p-2', 'p-3')
  .addE('VISITED')
  .to(V('c-1'))
  .id()
```

La requête doit insérer cinq arêtes : deux arêtes SUIVIES et trois arêtes VISITÉES. Cependant, la requête telle qu'elle est écrite insère huit arêtes : deux arêtes SUIVIES et six arêtes VISITÉES. Cela est dû au fait que l'opération qui insère les deux arêtes suivies émet deux traverseurs, ce qui entraîne l'exécution de l'opération suivante d'insertion de trois arêtes deux fois.

La solution consiste à ajouter une étape `fold()` après chaque opération susceptible d'émettre plusieurs traverseurs :

```
g.V('p-1')
  .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'p-1')
                          .property('email', 'person-1@example.org'))
  .V('p-2')
  .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'p-2').
                          .property('name', 'person-2@example.org'))
  .V('p-3')
  .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'p-3').
```

```
                        .property('name', 'person-3@example.org'))
.V('c-1')
.fold().
.coalesce(unfold(),
          addV('City').property(id, 'c-1').
                       .property('name', 'city-1'))
.V('p-1', 'p-2')
.addE('FOLLOWED')
.to(V('p-1'))
.fold()
.V('p-1', 'p-2', 'p-3')
.addE('VISITED')
.to(V('c-1')).
.id()
```

Nous avons inséré ici une étape `fold()` après l'opération qui insère les arêtes SUIVIES. Il en résulte un seul traverseur, et l'opération suivante n'est donc exécutée qu'une seule fois.

L'inconvénient de cette approche est que la requête n'est plus entièrement optimisée, car `fold()` n'est pas optimisé. L'opération d'insertion qui suit `fold()` ne sera maintenant pas optimisée non plus.

Si vous devez utiliser `fold()` pour réduire le nombre de traverseurs lors des étapes suivantes, essayez d'organiser les opérations de manière à ce que les moins coûteuses occupent la partie non optimisée de la requête.

Upserts qui modifient les sommets et les arêtes existants

Parfois, vous souhaitez créer un sommet ou une arête qui n'existe pas, puis y ajouter une propriété ou mettre à jour une propriété qui lui est associée, qu'il s'agisse d'un sommet ou d'une arête qui existe déjà ou pas encore.

Pour ajouter ou modifier une propriété, utilisez l'étape `property()`. Utilisez cette étape en dehors de l'étape `coalesce()`. Si vous essayez de modifier la propriété d'un sommet ou d'une arête qui existe déjà dans l'étape `coalesce()`, il est possible que la requête ne soit pas optimisée par le moteur de requêtes Neptune.

La requête suivante ajoute ou met à jour une propriété de compteur sur chaque sommet faisant l'objet d'un upsert. Chaque étape `property()` possède une cardinalité unique afin de garantir que les nouvelles valeurs remplacent toutes les valeurs existantes, plutôt que d'être ajoutées à un ensemble de valeurs existantes.

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-1')
                        .property('email', 'person-1@example.org'))
 .property(single, 'counter', 1)
 .V('v-2')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-2')
                        .property('email', 'person-2@example.org'))
 .property(single, 'counter', 2)
 .V('v-3')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-3')
                        .property('email', 'person-3@example.org'))
 .property(single, 'counter', 3)
 .id()
```

Si vous avez une valeur de propriété, telle qu'une valeur d'horodatage `lastUpdated`, qui s'applique à tous les éléments ajoutés par upsert, vous pouvez l'ajouter ou la mettre à jour à la fin de la requête :

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-1')
                        .property('email', 'person-1@example.org'))
 .V('v-2').
 .fold().
 .coalesce(unfold(),
          addV('Person').property(id, 'v-2')
                        .property('email', 'person-2@example.org'))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-3')
                        .property('email', 'person-3@example.org'))
 .V('v-1', 'v-2', 'v-3')
 .property(single, 'lastUpdated', datetime('2020-02-08'))
```

```
  .id()
```

Si d'autres conditions déterminent si un sommet ou une arête doit être encore modifié, vous pouvez utiliser une étape `has()` pour filtrer les éléments auxquels une modification sera appliquée. L'exemple suivant utilise une étape `has()` pour filtrer les sommets ajoutés par upsert en fonction de la valeur de leur propriété `version`. La requête fait ainsi passer à 3 la `version` de tout sommet dont l'élément `version` est inférieur à 3 :

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org')
                         .property('version', 3))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org')
                         .property('version', 3))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-3')
                         .property('email', 'person-3@example.org')
                         .property('version', 3))
 .V('v-1', 'v-2', 'v-3')
 .has('version', lt(3))
 .property(single, 'version', 3)
 .id()
```

Analyse de l'exécution des requêtes Neptune à l'aide de Gremlin (explication)

Amazon Neptune a ajouté une fonctionnalité Gremlin nommée explain. Cette fonctionnalité est un outil en libre-service qui vous aide à comprendre l'approche d'exécution adoptée par le moteur Neptune. Vous l'appelez en ajoutant un paramètre `explain` à un appel HTTP qui soumet une requête Gremlin.

La fonction `explain` fournit des informations sur la structure logique des plans d'exécution de requête. Vous pouvez utiliser ces informations pour identifier les goulots d'étranglement potentiels liés à l'évaluation et à l'exécution, comme expliqué dans Réglage des requêtes Gremlin. Vous pouvez aussi utiliser des indicateurs de requête pour améliorer les plans d'exécution de requêtes.

Rubriques

- Présentation du fonctionnement des requêtes Gremlin dans Neptune
- Utilisation de l'API Gremlin explain dans Neptune
- API de profil Garmlin dans Neptune
- Réglage des requêtes Gremlin à l'aide de la fonction Explique et du profil
- Prise en charge des étapes Gremlin natives dans Amazon Neptune

Présentation du fonctionnement des requêtes Gremlin dans Neptune

Pour tirer pleinement parti des rapports Gremlin `explain` et `profile` dans Amazon Neptune, il est utile de comprendre certaines informations contextuelles sur les requêtes Gremlin.

Rubriques

- Déclarations Gremlin dans Neptune
- Comment Neptune traite les requêtes Gremlin à l'aide d'index d'instruction
- Traitement des requêtes Gremlin dans Neptune

Déclarations Gremlin dans Neptune

Les données de graphe de propriétés d'Amazon Neptune sont composées de déclarations à quatre positions (quadruplets). Chacune de ces instructions représente une unité atomique individuelle des données de graphe de propriété. Pour de plus amples informations, veuillez consulter Modèle de données de graphe de Neptune. À l'instar du modèle de données RDF (Resource Description Framework), ces quatre positions sont les suivantes :

- `subject (S)`
- `predicate (P)`
- `object (O)`
- `graph (G)`

Chaque instruction est une assertion sur une ou plusieurs ressources. Par exemple, une instruction peut déclarer l'existence d'une relation entre deux ressources ou attacher une propriété (paire clé/valeur) à une ressource.

Vous pouvez considérer la position de prédicat (predicate) comme le verbe de l'instruction, qui décrit le type de relation ou de propriété. L'objet est la cible de la relation ou la valeur de la propriété. La position du graphe est facultative et peut être utilisée de nombreuses manières. Pour les données de graphe de propriétés (PG), elle est soit inutilisée (graphe null), soit utilisée pour représenter l'identifiant d'une arête. Un graphe est constitué d'un ensemble de déclarations avec des identifiants de ressources partagées.

Il existe trois classes de déclarations dans le modèle de données de graphe de propriétés Neptune :

Rubriques

- Instructions d'étiquette de sommet Gremlin
- Instructions d'arc Gremlin
- Instructions de propriété Gremlin

Instructions d'étiquette de sommet Gremlin

Les déclarations d'étiquette de sommet dans Neptune ont deux objectifs :

- Elles permettent d'assurer le suivi des étiquettes d'un sommet.
- La présence d'au moins une de ces instructions implique l'existence d'un sommet particulier dans le graphe.

Le sujet de ces instructions est un identifiant de sommet et l'objet est une étiquette, ces deux éléments étant spécifiés par l'utilisateur. Vous utilisez un prédicat fixe spécial pour ces instructions, affiché sous la forme `<~label>`, et un identifiant de graphe par défaut (graphe null), affiché sous la forme <~>.

Considérons par exemple la traversée `addV` suivante.

```
g.addV("Person").property(id, "v1")
```

Cette traversée entraîne l'ajout de l'instruction suivante au graphe.

```
StatementEvent[Added(<v1> <~label> <Person> <~>) .]
```

Instructions d'arc Gremlin

Une déclaration d'arête Gremlin implique l'existence d'une arête entre deux sommets dans un graphe dans Neptune. Le sujet (S) d'une instruction d'arc est le sommet source `from`. Le prédicat (P) est une étiquette d'arc fournie par l'utilisateur. L'objet (O) est le sommet cible `to`. Le graphe (G) est un identifiant d'arc fourni par l'utilisateur.

Considérons par exemple la traversée `addE` suivante.

```
g.addE("knows").from(V("v1")).to(V("v2")).property(id, "e1")
```

La traversée se traduit par l'ajout de l'instruction suivante au graphe.

```
StatementEvent[Added(<v1> <knows> <v2> <e1>) .]
```

Instructions de propriété Gremlin

Une déclaration de propriété Gremlin dans Neptune définit une valeur de propriété individuelle pour un sommet ou une arête. Le sujet est un identifiant d'arc ou de sommet fourni par l'utilisateur. Le prédicat est le nom de propriété (clé) et l'objet est la valeur de propriété individuelle. Le graphe (G) est à nouveau l'identifiant de graphe par défaut, le graphe null, affiché sous la forme <~>.

Prenons l'exemple de propriété de sommet suivant.

```
g.V("v1").property("name", "John")
```

Cette instruction aboutit aux résultats suivants.

```
StatementEvent[Added(<v1> <name> "John" <~>) .]
```

Les instructions de propriété diffèrent des autres en ce sens que leur objet est une valeur primitive (`string`, `date`, `byte`, `short`, `int`, `long`, `float` ou `double`). Leur objet n'est pas un identifiant de ressource pouvant être utilisé comme sujet d'une autre assertion.

Dans le cas de propriétés multiples, chaque valeur de propriété individuelle de l'ensemble reçoit sa propre instruction.

```
g.V("v1").property(set, "phone", "956-424-2563").property(set, "phone", "956-354-3692
 (tel:9563543692)")
```

La sortie obtenue est la suivante.

```
StatementEvent[Added(<v1> <phone> "956-424-2563" <~>) .]
StatementEvent[Added(<v1> <phone> "956-354-3692" <~>) .]
```

Les propriétés des arêtes sont gérées de la même manière que les propriétés des sommets, mais elles utilisent l'identifiant de bord en position (S). Par exemple, ajouter une propriété à une arête :

```
g.E("e1").property("weight", 0.8)
```

Cela entraîne l'ajout de l'instruction suivante au graphique.

```
StatementEvent[Added(<e1> <weight> 0.8 <~>) .]
```

Comment Neptune traite les requêtes Gremlin à l'aide d'index d'instruction

Les instructions sont accessibles dans Amazon Neptune au moyen de trois index d'instruction, comme indiqué dans Comment les instructions sont indexées dans Neptune. Neptune extrait un modèle d'instruction à partir d'une requête Gremlin dans laquelle certaines positions sont connues, tandis que les autres peuvent être découvertes via une recherche par index.

Neptune présume que le schéma de graphe de propriétés est de petite taille. Par conséquent, le nombre d'étiquettes d'arête et de noms de propriété distincts est relativement faible, ce qui se traduit par un faible nombre total de prédicats distincts. Neptune suit les prédicats distincts dans un index séparé. Il utilise ce cache de prédicats pour effectuer une analyse d'union de `{ all P x POGS }` au lieu d'utiliser un index OSGP. Le fait d'éviter d'avoir besoin d'un index OSGP de traversée inverse permet d'économiser de l'espace de stockage et du débit de charge.

L' Explain/Profile API Neptune Gremlin vous permet d'obtenir le nombre de prédicats dans votre graphique. Vous pouvez ensuite déterminer si votre application invalide l'hypothèse Neptune selon laquelle le schéma de votre graphe de propriété est de petite taille.

Les exemples suivants illustrent la façon dont Neptune utilise les index pour traiter les requêtes Gremlin.

Question : Quelles sont les étiquettes du sommet **v1** ?

```
  Gremlin code:      g.V('v1').label()
  Pattern:           (<v1>, <~label>, ?, ?)
  Known positions:   SP
```

```
Lookup positions:  OG
Index:             SPOG
Key range:         <v1>:<~label>:*
```

Question : Quelles sont les arêtes sortantes « knows » du sommet **v1** ?

```
Gremlin code:      g.V('v1').out('knows')
Pattern:           (<v1>, <knows>, ?, ?)
Known positions:   SP
Lookup positions:  OG
Index:             SPOG
Key range:         <v1>:<knows>:*
```

Question : Quels sont les sommets qui ont une étiquette de sommet **`Person`** ?

```
Gremlin code:      g.V().hasLabel('Person')
Pattern:           (?, <~label>, <Person>, <~>)
Known positions:   POG
Lookup positions:  S
Index:             POGS
Key range:         <~label>:<Person>:<~>:*
```

Question : Quels sont les from/to sommets d'une arête donnée ? **`e1`**

```
Gremlin code:      g.E('e1').bothV()
Pattern:           (?, ?, ?, <e1>)
Known positions:   G
Lookup positions:  SPO
Index:             GPSO
Key range:         <e1>:*
```

Neptune ne possède pas d'index de déclarations tel qu'un index OSGP de traversée inverse. Ce type d'index peut être utilisé pour collecter toutes les arêtes entrantes sur toutes les étiquettes d'arête, comme dans l'exemple suivant.

Question : Quels sont les sommets adjacents entrants **v1** ?

```
Gremlin code:      g.V('v1').in()
Pattern:           (?, ?, <v1>, ?)
Known positions:   O
Lookup positions:  SPG
```

```
Index:              OSGP  // <-- Index does not exist
```

Traitement des requêtes Gremlin dans Neptune

Dans Amazon Neptune, les traversées plus complexes peuvent être représentées par une série de modèles qui créent une relation basée sur la définition de variables nommées pouvant être partagées entre des modèles pour créer des jointures. Voici un exemple :

Question : Quel est le voisinage à deux sauts du sommet **v1** ?

```
Gremlin code:      g.V('v1').out('knows').out('knows').path()
Pattern:           (?1=<v1>, <knows>, ?2, ?) X Pattern(?2, <knows>, ?3, ?)

The pattern produces a three-column relation (?1, ?2, ?3) like this:
                   ?1     ?2     ?3
                   ================
                   v1     v2     v3
                   v1     v2     v4
                   v1     v5     v6
```

En partageant la variable ?2 entre les deux modèles (à la position O dans le premier modèle et à la position S dans le second modèle), vous créez une jointure entre les voisins à un premier saut et les voisins à deux sauts. Chaque solution Neptune possède des liaisons pour les trois variables nommées, qui peuvent être utilisées pour recréer un TinkerPopTraverser (y compris les informations de chemin).

La première étape du traitement d'une requête Gremlin consiste à analyser la requête pour en faire un objet TinkerPop Traversal, composé d'une série d'étapes. TinkerPop Ces étapes, qui font partie du TinkerPop projet open source Apache, sont à la fois les opérateurs logiques et physiques qui composent une traversée Gremlin dans l'implémentation de référence. Elles sont toutes deux utilisées pour représenter le modèle de la requête. Il s'agit d'opérateurs exécutables qui peuvent produire des solutions en fonction de la sémantique de l'opérateur qu'ils représentent. Par exemple, `.V()` est à la fois représenté et exécuté par le TinkerPop GraphStep.

Comme ces TinkerPop étapes prêtes à l'emploi sont exécutables, un tel TinkerPop Traversal peut exécuter n'importe quelle requête Gremlin et produire la bonne réponse. Cependant, lorsqu'elles sont exécutées sur un graphique de grande taille, TinkerPop les étapes peuvent parfois être très inefficaces et lentes. Au lieu de les utiliser, Neptune essaie alors de convertir la traversée en forme déclarative composée de groupes de modèles, comme décrit précédemment.

Neptune ne prend actuellement pas en charge tous les opérateurs Gremlin (étapes) dans son moteur de requête natif. Il essaie donc de réduire autant d'étapes que possible en une seule étape `NeptuneGraphQueryStep`, qui contient le plan de requête logique déclaratif pour toutes les étapes qui ont été converties. Idéalement, toutes les étapes sont converties. Mais lorsqu'une étape ne peut pas être convertie, Neptune sort de l'exécution native et reporte toute exécution de requête à partir de ce point jusqu'aux étapes. TinkerPop Il n'essaie pas de jongler avec et sans l'exécution native.

Une fois que les étapes sont converties en plan de requête logique, Neptune exécute une série d'optimiseurs de requête qui réécrivent le plan de requête en fonction de l'analyse statique et des cardinalités estimées. Ces optimiseurs effectuent des opérations telles que la réorganisation des opérateurs en fonction du nombre de plages, la suppression des opérateurs superflus ou redondants, la réorganisation des filtres, le transfert des opérateurs dans différents groupes, etc.

Une fois qu'un plan de requête optimisé est généré, Neptune crée un pipeline d'opérateurs physiques qui effectuent le travail d'exécution de la requête. Cela inclut la lecture des données à partir des index d'instruction, l'exécution de jointures de différents types, le filtrage, l'organisation, etc. Le pipeline produit un flux de solution qui est ensuite reconverti en un flux d'objets TinkerPop Traverser.

Sérialisation des résultats de requête

Amazon Neptune s'appuie actuellement sur les sérialiseurs de messages de TinkerPop réponse pour convertir les résultats des requêtes (TinkerPop Traversers) en données sérialisées à renvoyer par câble au client. Ces formats de sérialisation ont tendance à être assez verbeux.

Par exemple, pour sérialiser le résultat d'une requête de sommet telle que `g.V().limit(1)`, le moteur de requête Neptune doit effectuer une seule recherche pour générer le résultat. Toutefois, le sérialiseur `GraphSON` effectue un grand nombre de recherches supplémentaires pour empaqueter le sommet dans le format de sérialisation. Il doit effectuer une recherche pour obtenir l'étiquette, une recherche pour obtenir les clés de propriété et une recherche par clé de propriété pour que le sommet puisse obtenir toutes les valeurs de chaque clé.

Certains formats de sérialisation sont plus efficaces, mais tous nécessitent des recherches supplémentaires. De plus, les TinkerPop sérialiseurs n'essaient pas d'éviter les recherches dupliquées, ce qui entraîne souvent la répétition inutile de nombreuses recherches.

Il est donc très important d'écrire vos requêtes afin qu'elles demandent spécifiquement les informations dont elles ont besoin. Par exemple, `g.V().limit(1).id()` renvoie uniquement l'ID de sommet et élimine toutes les recherches de sérialiseur supplémentaires. L'[API de profil Garmlin dans Neptune] vous permet de voir combien d'appels de recherche sont effectués pendant l'exécution de la requête et pendant la sérialisation.

Utilisation de l'API Gremlin **`explain`** dans Neptune

L'API Amazon Neptune Gremlin `explain` renvoie le plan de requête qui serait exécuté si une requête spécifique était exécutée. Comme l'API n'exécute pas réellement la requête, le plan est renvoyé presque instantanément.

Elle diffère de l'étape TinkerPop .explain () afin de pouvoir rapporter des informations spécifiques au moteur Neptune.

Informations contenues dans un rapport d'**`explication`** de Gremlin

Un rapport `explain` contient les informations suivantes :

- Chaîne de requête demandée.
- Traversée d'origine. Il s'agit de l'objet TinkerPop Traversal produit en analysant la chaîne de requête en TinkerPop étapes. Elle est équivalente à la requête d'origine produite en exécutant la requête `.explain()` sur le TinkerPop TinkerGraph.
- Traversée convertie. Il s'agit du Neptune Traversal produit en convertissant le TinkerPop Traversal en représentation logique du plan de requêtes Neptune. Dans de nombreux cas, l'intégralité de la TinkerPop traversée est convertie en deux étapes Neptune : l'une qui exécute la requête complète `NeptuneGraphQueryStep` () et l'autre qui reconvertit la sortie du moteur de requêtes Neptune en Traversers (). TinkerPop `NeptuneTraverserConverterStep`
- Traversée optimisée. Il s'agit de la version optimisée du plan de requête Neptune une fois qu'il a été exécuté via une série d'optimiseurs statiques de réduction du travail qui réécrivent la requête en fonction de l'analyse statique et des cardinalités estimées. Ces optimiseurs effectuent des opérations telles que la réorganisation des opérateurs en fonction du nombre de plages, la suppression des opérateurs superflus ou redondants, la réorganisation des filtres, le transfert des opérateurs dans différents groupes, etc.
- Nombre de prédicats. En raison de la stratégie d'indexation Neptune décrite précédemment, le fait d'avoir un grand nombre de prédicats différents peut entraîner des problèmes de performances. Cela est particulièrement vrai pour les requêtes qui utilisent des opérateurs de traversée inverse sans étiquette d'arc (`.in` ou `.both`). Si ces opérateurs sont utilisés et que le nombre de prédicats est suffisamment élevé, le rapport `explain` affiche un message d'avertissement.
- Informations DFE. Lorsque le moteur alternatif DFE est activé, les composants de traversée suivants peuvent apparaître dans la traversée optimisée :
  - **`DFEStep`** : étape DFE optimisée pour Neptune dans la traversée qui contient un objet `DFENode` enfant. `DFEStep` représente la partie du plan de requête exécutée dans le moteur DFE.

- **`DFENode`** : contient la représentation intermédiaire sous la forme d'un ou de plusieurs objets `DFEJoinGroupNodes` enfants.
- **`DFEJoinGroupNode`** : représente une jointure d'un ou de plusieurs éléments `DFENode` ou `DFEJoinGroupNode`.
- **`NeptuneInterleavingStep`** : étape DFE optimisée pour Neptune dans la traversée qui contient un objet `DFEStep` enfant.

  Contient également un élément `stepInfo` comportant des informations sur la traversée, telles que l'élément frontière, les éléments de chemin utilisés, etc. Ces informations sont utilisées pour traiter l'objet `DFEStep` enfant.

Un moyen simple de savoir si votre requête est évaluée par le DFE est de vérifier si la sortie `explain` contient un objet `DFEStep`. Toute partie de la traversée qui n'en fait pas partie ne `DFEStep` sera pas exécutée par DFE et sera exécutée par le TinkerPop moteur.

Consultez Exemple avec le DFE activé pour voir un exemple de rapport.

Gremlin **`explique`** la syntaxe

La syntaxe de l'`explain`API est la même que celle de l'API HTTP pour les requêtes, sauf qu'elle est utilisée `/gremlin/explain` comme point de terminaison au lieu de`/gremlin`, comme dans les exemples suivants.

AWS CLI

```
aws neptunedata execute-gremlin-explain-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --gremlin-query "g.V().limit(1)"
```

Pour plus d'informations, consultez execute-gremlin-explain-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
```

```
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_gremlin_explain_query(
    gremlinQuery='g.V().limit(1)'
)

print(response['output'])
```

Pour des exemples de AWS SDK dans d'autres langages tels que Java, .NET, etc., consultezAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/gremlin/explain \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d '{"gremlin":"g.V().limit(1)"}'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultezUtiliser `awscurl` avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

curl

```
curl -X POST https://your-neptune-endpoint:port/gremlin/explain \
  -d '{"gremlin":"g.V().limit(1)"}'
```

La requête précédente génèrerait la sortie suivante.

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************
```

```
Query String
============
g.V().limit(1)

Original Traversal
==================
[GraphStep(vertex,[]), RangeGlobalStep(0,1)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
        }, finishers=[limit(1)], annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, finishers=[limit(1)], annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 18
```

Étapes non converties TinkerPop

Idéalement, toutes les TinkerPop étapes d'une traversée bénéficient d'une couverture native de l'opérateur Neptune. Lorsque ce n'est pas le cas, Neptune recourt à l'exécution par TinkerPop étapes pour combler les lacunes dans la couverture de ses opérateurs. Si une traversée utilise une

étape pour laquelle Neptune n'a pas encore de couverture native, le rapport `explain` affiche un avertissement indiquant où l'écart s'est produit.

Lorsqu'une étape sans opérateur Neptune natif correspondant est rencontrée, la totalité de la traversée à partir de ce point est exécutée par TinkerPop étapes, même si les étapes suivantes comportent des opérateurs Neptune natifs.

L'exception est lorsque la recherche en texte intégral Neptune est appelée. Il NeptuneSearchStep implémente des étapes sans équivalents natifs sous forme d'étapes de recherche en texte intégral.

Exemple de sortie **`Explique`** où toutes les étapes d'une requête ont des équivalents natifs

Voici un exemple de rapport `explain` concernant une requête pour laquelle toutes les étapes ont des équivalents natifs :

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============
g.V().out()

Original Traversal
==================
[GraphStep(vertex,[]), VertexStep(OUT,vertex)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
```

```
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 18
```

Exemple où certaines étapes d'une requête n'ont pas d'équivalents natifs

Neptune gère `GraphStep` et `VertexStep` en mode natif, mais si vous introduisez un élément `FoldStep` et un élément `UnfoldStep` , la sortie `explain` générée est différente :

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============
g.V().fold().unfold().out()

Original Traversal
==================
[GraphStep(vertex,[]), FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]
```

```
+ not converted into Neptune steps: [FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

Optimized Traversal
==================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep,
    NeptuneMemoryTrackerStep
]
+ not converted into Neptune steps: [FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

WARNING: >> FoldStep << is not supported natively yet
```

Dans ce cas, l'élément `FoldStep` vous fait quitter l'exécution native. Même l'étape `VertexStep` suivante n'est plus gérée en mode natif, car elle apparaît en aval des étapes `Fold/Unfold`.

Pour des raisons de performances et de réduction des coûts, il est important que vous essayiez de formuler des traversées de manière à ce que le maximum de travail possible soit effectué de manière native dans le moteur de requêtes Neptune, plutôt que de procéder à des implémentations étape par étape. TinkerPop

Exemple d'une requête qui utilise la recherche en texte intégral Neptune

La requête suivante utilise la recherche en texte intégral Neptune :

```
g.withSideEffect("Neptune#fts.endpoint", "some_endpoint")
  .V()
  .tail(100)
  .has("Neptune#fts mark*")
  -------
  .has("name", "Neptune#fts mark*")
  .has("Person", "name", "Neptune#fts mark*")
```

La partie `.has("name", "Neptune#fts mark*")` limite la recherche aux sommets avec `name`, alors que `.has("Person", "name", "Neptune#fts mark*")` limite la recherche aux sommets avec `name` et l'étiquette `Person`. Le résultat est la traversée suivante dans le rapport `explain` :

```
Final Traversal
[NeptuneGraphQueryStep(Vertex) {
    JoinGroupNode {
        PatternNode[(?1, termid(1,URI), ?2, termid(0,URI)) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
    }, annotations={path=[Vertex(?1):GraphStep], maxVarId=4}
}, NeptuneTraverserConverterStep, NeptuneTailGlobalStep(10),
 NeptuneTinkerpopTraverserConverterStep, NeptuneSearchStep {
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=some_endpoint}
    }
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=some_endpoint}
    }
}]
```

Exemple d'utilisation : **`expliquer`** quand le DFE est activé

Voici un exemple de rapport `explain` lorsque le moteur de requête alternatif DFE est activé :

```
********************************************************
                Neptune Gremlin Explain
********************************************************

Query String
============

g.V().as("a").out().has("name", "josh").out().in().where(eq("a"))


Original Traversal
==================
[GraphStep(vertex,[])@[a], VertexStep(OUT,vertex), HasStep([name.eq(josh)]),
 VertexStep(OUT,vertex), VertexStep(IN,vertex), WherePredicateStep(eq(a))]

Converted Traversal
===================
Neptune steps:
[
    DFEStep(Vertex) {
      DFENode {
```

```
        DFEJoinGroupNode[ children={
          DFEPatternNode[(?1, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, ?2,
 <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>) . project DISTINCT[?1]
 {rangeCountEstimate=unknown}],
          DFEPatternNode[(?1, ?3, ?4, ?5) . project ALL[?1, ?4] graphFilters=(!
= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}]
        }, {rangeCountEstimate=unknown}
        ]
      } [Vertex(?1):GraphStep@[a], Vertex(?4):VertexStep]
    } ,
    NeptuneTraverserConverterDFEStep
]
+ not converted into Neptune steps: HasStep([name.eq(josh)]),
Neptune steps:
[
    NeptuneInterleavingStep {
      StepInfo[joinVars=[?7, ?1], frontierElement=Vertex(?7):HasStep,
 pathElements={a=(last,Vertex(?1):GraphStep@[a])}, listPathElement={}, indexTime=0ms],
      DFEStep(Vertex) {
        DFENode {
          DFEJoinGroupNode[ children={
            DFEPatternNode[(?7, ?8, ?9, ?10) . project ALL[?7, ?9]
 graphFilters=(!= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}],
            DFEPatternNode[(?12, ?11, ?9, ?13) . project ALL[?9, ?12]
 graphFilters=(!= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}]
          }, {rangeCountEstimate=unknown}
          ]
        } [Vertex(?9):VertexStep, Vertex(?12):VertexStep]
      }
    }
]
+ not converted into Neptune steps: WherePredicateStep(eq(a)),
Neptune steps:
[
    DFECleanupStep
]


Optimized Traversal
===================
Neptune steps:
```

```
[
    DFEStep(Vertex) {
      DFENode {
        DFEJoinGroupNode[ children={
          DFEPatternNode[(?1, ?3, ?4, ?5) . project ALL[?1, ?4] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807}]
        }, {rangeCountEstimate=unknown}
        ]
      } [Vertex(?1):GraphStep@[a], Vertex(?4):VertexStep]
    } ,
    NeptuneTraverserConverterDFEStep
]
+ not converted into Neptune steps: NeptuneHasStep([name.eq(josh)]),
Neptune steps:
[
    NeptuneMemoryTrackerStep,
    NeptuneInterleavingStep {
      StepInfo[joinVars=[?7, ?1], frontierElement=Vertex(?7):HasStep,
 pathElements={a=(last,Vertex(?1):GraphStep@[a])}, listPathElement={}, indexTime=0ms],
      DFEStep(Vertex) {
        DFENode {
          DFEJoinGroupNode[ children={
            DFEPatternNode[(?7, ?8, ?9, ?10) . project ALL[?7, ?9] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807}],
            DFEPatternNode[(?12, ?11, ?9, ?13) . project ALL[?9, ?12] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807}]
          }, {rangeCountEstimate=unknown}
          ]
        } [Vertex(?9):VertexStep, Vertex(?12):VertexStep]
      }
    }
]
+ not converted into Neptune steps: WherePredicateStep(eq(a)),
Neptune steps:
[
    DFECleanupStep
]


WARNING: >> [NeptuneHasStep([name.eq(josh)]), WherePredicateStep(eq(a))] << (or one of
 the children for each step) is not supported natively yet

Predicates
==========
```

```
# of predicates: 8
```

Voir Informations dans `explain` pour une description des DFE-specific sections du rapport.

API de **`profil Garmlin`** dans Neptune

L'API Neptune Gremlin `profile` effectue une traversée Gremlin spécifique, collecte différentes métriques sur l'exécution et génère en sortie un rapport de profil.

Elle diffère de l'étape TinkerPop .profile () afin de pouvoir rapporter des informations spécifiques au moteur Neptune.

Le rapport de profil inclut les informations suivantes concernant le plan de requête :

- Le pipeline de l'opérateur physique
- Les opérations d'index pour l'exécution et la sérialisation des requêtes
- La taille du résultat

L'API `profile` utilise une version étendue de la syntaxe d'API HTTP pour les requêtes, avec `/gremlin/profile` comme point de terminaison au lieu de `/gremlin`.

Paramètres spécifiques au profil de Neptune Gremlin

- profile.results – `boolean`, valeurs autorisées : `TRUE` et `FALSE`, valeur par défaut : `TRUE`.

  Si le paramètre est défini true, les résultats de la requête sont collectés et affichés dans le rapport `profile`. Si le paramètre a pour valeur false, seul le nombre de résultats s'affiche.
- profile.chop – `int`, valeur par défaut : 250.

  Si la valeur du paramètre est différente de zéro, la chaîne de résultats est tronquée à ce nombre de caractères. Cela ne empêche pas la capture de tous les résultats. Cela limite simplement la taille de la chaîne dans le rapport de profil. Si la valeur est zéro, la chaîne contient tous les résultats.
- profile.serializer – `string`, valeur par défaut : `<null>`.

  Si la valeur du paramètre est différente de null, les résultats collectés sont renvoyés dans un message de réponse sérialisé au format spécifié par ce paramètre. Le nombre d'opérations d'index nécessaires pour générer ce message de réponse est signalé, ainsi que la taille en octets à envoyer au client.

Les valeurs autorisées sont <null> ou n'importe laquelle des valeurs d'énumération valides du type MIME ou TinkerPop du pilote « Serializers ».

```
"application/json" or "GRAPHSON"
"application/vnd.gremlin-v1.0+json" or "GRAPHSON_V1"
"application/vnd.gremlin-v1.0+json;types=false" or "GRAPHSON_V1_UNTYPED"
"application/vnd.gremlin-v2.0+json" or "GRAPHSON_V2"
"application/vnd.gremlin-v2.0+json;types=false" or "GRAPHSON_V2_UNTYPED"
"application/vnd.gremlin-v3.0+json" or "GRAPHSON_V3"
"application/vnd.gremlin-v3.0+json;types=false" or "GRAPHSON_V3_UNTYPED"
"application/vnd.graphbinary-v1.0" or "GRAPHBINARY_V1"
```

- profile.indexOps – `boolean`, valeurs autorisées : `TRUE` et `FALSE`, valeur par défaut : `FALSE`.

  Si ce paramètre a la valeur true, il affiche un rapport détaillé de toutes les opérations d'index qui ont eu lieu au cours de l'exécution et de la sérialisation des requêtes. Avertissement : ce rapport peut être très détaillé.

Exemple de sortie du profil de Neptune Garmlin

Voici un exemple de requête `profile`.

AWS CLI

```
aws neptunedata execute-gremlin-profile-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --gremlin-query 'g.V().hasLabel("airport").has("code",
 "AUS").emit().repeat(in().simplePath()).times(2).limit(100)' \
  --serializer "application/vnd.gremlin-v3.0+json"
```

Pour plus d'informations, consultez execute-gremlin-profile-query dans le manuel de référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
```

```
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_gremlin_profile_query(
    gremlinQuery='g.V().hasLabel("airport").has("code",
 "AUS").emit().repeat(in().simplePath()).times(2).limit(100)',
    serializer='application/vnd.gremlin-v3.0+json'
)

print(response['output'])
```

Pour des exemples de AWS SDK dans d'autres langages tels que Java, .NET, etc., consultezAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/gremlin/profile \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d '{"gremlin":"g.V().hasLabel(\"airport\").has(\"code\",
 \"AUS\").emit().repeat(in().simplePath()).times(2).limit(100)",
 "profile.serializer":"application/vnd.gremlin-v3.0+json"}'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultezUtiliser `awscurl` avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

curl

```
curl -X POST https://your-neptune-endpoint:port/gremlin/profile \
  -d '{"gremlin":"g.V().hasLabel(\"airport\").has(\"code\",
 \"AUS\").emit().repeat(in().simplePath()).times(2).limit(100)",
 "profile.serializer":"application/vnd.gremlin-v3.0+json"}'
```

Cette requête génère le rapport `profile` suivant lorsqu'elle est exécutée sur l'exemple de graphe des lignes aériennes provenant du billet de blog, Let Me Graph That For You - Part 1 - Air Routes.

```
********************************************************
                Neptune Gremlin Profile
********************************************************

Query String
==================
g.V().hasLabel("airport").has("code",
 "AUS").emit().repeat(in().simplePath()).times(2).limit(100)

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([~label.eq(airport), code.eq(AUS)]),
 RepeatStep(emit(true),[VertexStep(IN,vertex), PathFilterStep(simple),
 RepeatEndStep],until(loops(2))), RangeGlobalStep(0,100)]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "AUS", ?) . project ?1 .],
 {estimatedCardinality=1, indexTime=84, hashJoin=true, joinTime=3, actualTotalOutput=1}
            PatternNode[(?1, <~label>, ?2=<airport>, <~>) . project ask .],
 {estimatedCardinality=3374, indexTime=29, hashJoin=true, joinTime=0,
 actualTotalOutput=61}
            RepeatNode {
                Repeat {
                    PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .
 SimplePathFilter(?1, ?3)) .], {hashJoin=true, estimatedCardinality=50148, indexTime=0,
 joinTime=3}
                }
                Emit {
                    Filter(true)
                }
                LoopsCondition {
                    LoopsFilter([?1, ?3],eq(2))
                }
            }, annotations={repeatMode=BFS, emitFirst=true, untilFirst=false, leftVar=?
1, rightVar=?3}
```

```
        }, finishers=[limit(100)], annotations={path=[Vertex(?1):GraphStep,
 Repeat[Vertex(?3):VertexStep]], joinStats=true, optimizationTime=495, maxVarId=7,
 executionTime=323}
    },
    NeptuneTraverserConverterStep
]

Physical Pipeline
=================
NeptuneGraphQueryStep
    |-- StartOp
    |-- JoinGroupOp
        |-- SpoolerOp(100)
        |-- DynamicJoinOp(PatternNode[(?1, <code>, "AUS", ?) . project ?1 .],
 {estimatedCardinality=1, indexTime=84, hashJoin=true})
        |-- SpoolerOp(100)
        |-- DynamicJoinOp(PatternNode[(?1, <~label>, ?2=<airport>, <~>) . project
 ask .], {estimatedCardinality=3374, indexTime=29, hashJoin=true})
        |-- RepeatOp
            |-- <upstream input> (Iteration 0) [visited=1, output=1 (until=0, emit=1),
 next=1]
            |-- BindingSetQueue (Iteration 1) [visited=61, output=61 (until=0,
 emit=61), next=61]
                |-- SpoolerOp(100)
                |-- DynamicJoinOp(PatternNode[(?3, ?5, ?1, ?6) . project ?
1,?3 . IsEdgeIdFilter(?6) . SimplePathFilter(?1, ?3)) .], {hashJoin=true,
 estimatedCardinality=50148, indexTime=0})
            |-- BindingSetQueue (Iteration 2) [visited=38, output=38 (until=38,
 emit=0), next=0]
                |-- SpoolerOp(100)
                |-- DynamicJoinOp(PatternNode[(?3, ?5, ?1, ?6) . project ?
1,?3 . IsEdgeIdFilter(?6) . SimplePathFilter(?1, ?3)) .], {hashJoin=true,
 estimatedCardinality=50148, indexTime=0})
        |-- LimitOp(100)

Runtime (ms)
============
Query Execution:  392.686
Serialization:   2636.380

Traversal Metrics
=================
Step                                                               Count  Traversers
    Time (ms)    % Dur
```

```
-------------------------------------------------------------------------------------------------------
NeptuneGraphQueryStep(Vertex)                                          100         100
      314.162    82.78
NeptuneTraverserConverterStep                                          100         100
       65.333    17.22
                                          >TOTAL                         -           -
      379.495        -

Repeat Metrics
==============
Iteration  Visited   Output    Until     Emit     Next
----------------------------------------------------------
        0        1        1        0        1        1
        1       61       61        0       61       61
        2       38       38       38        0        0
----------------------------------------------------------
               100      100       38       62       62

Predicates
==========
# of predicates: 16

WARNING: reverse traversal with no edge label(s) - .in() / .both() may impact query
 performance

Results
=======
Count: 100
Output: [v[3], v[3600], v[3614], v[4], v[5], v[6], v[7], v[8], v[9], v[10], v[11],
 v[12], v[47], v[49], v[136], v[13], v[15], v[16], v[17], v[18], v[389], v[20], v[21],
 v[22], v[23], v[24], v[25], v[26], v[27], v[28], v[416], v[29], v[30], v[430], v[31],
 v[9...
Response serializer: GRYO_V3D0
Response size (bytes): 23566

Index Operations
================
Query execution:
    # of statement index ops: 3
    # of unique statement index ops: 3
    Duplication ratio: 1.0
    # of terms materialized: 0
Serialization:
    # of statement index ops: 200
```

```
    # of unique statement index ops: 140
    Duplication ratio: 1.43
    # of terms materialized: 393
```

Outre les plans de requête renvoyés par un appel à Neptune `explain`, les résultats de la requête `profile` incluent des statistiques sur l'exécution des requêtes. Chaque opération de jointure est balisée avec le temps nécessaire à l'exécution de sa jointure, ainsi que le nombre réel de solutions qui lui ont été transmises.

La sortie de `profile` inclut le temps nécessaire à la phase d'exécution de la requête principale, ainsi qu'à la phase de sérialisation si l'option `profile.serializer` a été spécifiée.

La répartition des opérations d'index effectuées au cours de chaque phase est également incluse au bas de la sortie `profile`.

Notez que les exécutions consécutives d'une même requête peuvent afficher des résultats différents en termes d'opérations d'exécution et d'index en raison de la mise en cache.

Pour les requêtes utilisant l'étape `repeat()`, une répartition de la frontière sur chaque itération est disponible si l'étape `repeat()` a été réduite et intégrée dans une étape `NeptuneGraphQueryStep`.

Différences entre les rapports **`de profil`** lorsque le DFE est activé

Lorsque le moteur de requête alternatif DFE Neptune est activé, la sortie `profile` est quelque peu différente :

Traversée optimisée : cette section est similaire à celle de la sortie `explain`, mais contient des informations supplémentaires. Cela inclut le type d'opérateurs DFE pris en compte lors de la planification, ainsi que les estimations de coûts associées au pire et au meilleur scénario.

Pipeline physique : cette section capture les opérateurs utilisés pour exécuter la requête. Les éléments `DFESubQuery` procèdent à l'abstraction du plan physique utilisé par le DFE pour exécuter la partie du plan dont il est responsable. Les éléments `DFESubQuery` sont présentés dans la section suivante où les statistiques DFE sont répertoriées.

DFEQueryEngine Statistiques : cette section n'apparaît que lorsqu'au moins une partie de la requête est exécutée par DFE. Elle décrit diverses statistiques d'exécution spécifiques au DFE et contient une répartition détaillée du temps passé dans les différentes parties de l'exécution de la requête, par `DFESubQuery`.

Les sous-requêtes imbriquées dans différents éléments `DFESubQuery` ne sont pas hiérarchisées dans cette section, et les identifiants uniques sont marqués d'un en-tête commençant par `subQuery=`.

Mesures de traversée : cette section présente les mesures de traversée au niveau des étapes et, lorsque le moteur DFE exécute la totalité ou une partie de la requête, affiche les mesures pour. `DFEStep` and/or `NeptuneInterleavingStep` Consultez Réglage des requêtes Gremlin à l'aide de la fonction Explique et du profil.

> **Note**
>
> La prise en charge du format DFE pour G705 étant une fonctionnalité expérimentale, le format exact de la `profile` sortie est sujet à modification.

Exemple **`de sortie de profil`** lorsque le moteur Neptune Dataflow (DFE) est activé

Lorsque le moteur DFE est utilisé pour exécuter des requêtes Gremlin, la sortie de l'API Gremlin `profile` est mise en forme comme indiqué dans l'exemple ci-dessous.

Interrogation :

AWS CLI

```
aws neptunedata execute-gremlin-profile-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --gremlin-query "g.withSideEffect('Neptune#useDFE', true).V().has('code',
 'ATL').out()"
```

Pour plus d'informations, consultez execute-gremlin-profile-query dans le manuel de référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)
```

```
response = client.execute_gremlin_profile_query(
    gremlinQuery="g.withSideEffect('Neptune#useDFE', true).V().has('code',
 'ATL').out()"
)

print(response['output'])
```

Pour des exemples de AWS SDK dans d'autres langages tels que Java, .NET, etc., consultezAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/gremlin/profile \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d '{"gremlin":"g.withSideEffect('"'"'Neptune#useDFE'"'"',
 true).V().has('"'"'code'"'"', '"'"'ATL'"'"').out()"}'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultezUtiliser `awscurl` avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

curl

```
curl -X POST https://your-neptune-endpoint:port/gremlin/profile \
  -d '{"gremlin":"g.withSideEffect('"'"'Neptune#useDFE'"'"',
 true).V().has('"'"'code'"'"', '"'"'ATL'"'"').out()"}'
```

```
*******************************************************
                    Neptune Gremlin Profile
    *****************************************************

    Query String
```

```
    ==================
    g.withSideEffect('Neptune#useDFE', true).V().has('code', 'ATL').out()

    Original Traversal
    ==================
    [GraphStep(vertex,[]), HasStep([code.eq(ATL)]), VertexStep(OUT,vertex)]

    Optimized Traversal
    ===================
    Neptune steps:
    [
        DFEStep(Vertex) {
          DFENode {
            DFEJoinGroupNode[null](
              children=[
                DFEPatternNode((?1, vp://code[419430926], ?4, defaultGraph[526]) .
project DISTINCT[?1] objectFilters=(in(ATL[452987149]) . ), {rangeCountEstimate=1},
                  opInfo=(type=PipelineJoin,
cost=(exp=(in=1.00,out=1.00,io=0.00,comp=0.00,mem=0.00),wc=(in=1.00,out=1.00,io=0.00,comp=0.00
                    disc=(type=PipelineScan,
cost=(exp=(in=1.00,out=1.00,io=0.00,comp=0.00,mem=34.00),wc=(in=1.00,out=1.00,io=0.00,comp=0.0
                DFEPatternNode((?1, ?5, ?6, ?7) . project ALL[?1, ?6] graphFilters=(!=
defaultGraph[526] . ), {rangeCountEstimate=92233720368547758O7})],
              opInfo=[
                OperatorInfoWithAlternative[
                  rec=(type=PipelineJoin,
cost=(exp=(in=1.00,out=27.76,io=0.00,comp=0.00,mem=0.00),wc=(in=1.00,out=27.76,io=0.00,comp=0.
                    disc=(type=PipelineScan,
cost=(exp=(in=1.00,out=27.76,io=Infinity,comp=0.00,mem=295147905179352830000.00),wc=(in=1.00,c
                  alt=(type=PipelineScan,
cost=(exp=(in=1.00,out=27.76,io=Infinity,comp=0.00,mem=295147905179352830000.00),wc=(in=1.00,c
          } [Vertex(?1):GraphStep, Vertex(?6):VertexStep]
        } ,
        NeptuneTraverserConverterDFEStep,
        DFECleanupStep
    ]


    Physical Pipeline
    =================
    DFEStep
        |-- DFESubQuery1

    DFEQueryEngine Statistics
```

```
    =================
    DFESubQuery1

 ########################################################################################################
    # ID # Out #1 # Out #2 # Name                  # Arguments
                                                                                          # Mode #
 Units In # Units Out # Ratio  # Time (ms) #

 ########################################################################################################
    # 0  # 1      # -      # DFESolutionInjection  # solutions=[]
                                                                                       # -    # 0
      # 1         # 0.00   # 0.01      #
    #    #        #        #                       # outSchema=[]
                                                                                       #      #
      #           #        #           #

 ########################################################################################################
    # 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_1 # -    #
 1        # 1         # 1.00   # 0.02      #

 ########################################################################################################
    # 2  # 3      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_2 # -    #
 1        # 242       # 242.00 # 0.02      #

 ########################################################################################################
    # 3  # 4      # -      # DFEMergeChunks        # -
                                                                                       # -    # 242
      # 242       # 1.00   # 0.01      #

 ########################################################################################################
    # 4  # -      # -      # DFEDrain              # -
                                                                                       # -    # 242
      # 0         # 0.00   # 0.01      #

 ########################################################################################################


    subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/
graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_1

 ########################################################################################################
```

```
   # ID # Out #1 # Out #2 # Name                 # Arguments
                        # Mode # Units In # Units Out # Ratio # Time (ms) #

 ######################################################################################################
   # 0  # 1      # -      # DFEPipelineScan      # pattern=Node(?1) with property
 'code' as ?4 and label 'ALL' # -    # 0        # 1         # 0.00  # 0.22      #
   #    #        #        #                     # inlineFilters=[(?4 IN ["ATL"])]
                        #      #          #           #       #           #
   #    #        #        #                     # patternEstimate=1
                        #      #          #           #       #           #

 ######################################################################################################
   # 1  # 2      # -      # DFEMergeChunks       # -
                        # -    # 1        # 1         # 1.00  # 0.02      #

 ######################################################################################################
   # 2  # 4      # -      # DFERelationalJoin    # joinVars=[]
                        # -    # 2        # 1         # 0.50  # 0.09      #

 ######################################################################################################
   # 3  # 2      # -      # DFESolutionInjection # solutions=[]
                        # -    # 0        # 1         # 0.00  # 0.01      #
   #    #        #        #                     # outSchema=[]
                        #      #          #           #       #           #

 ######################################################################################################
   # 4  # -      # -      # DFEDrain             # -
                        # -    # 1        # 0         # 0.00  # 0.01      #

 ######################################################################################################


   subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/
graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_2

 ######################################################################################################
   # ID # Out #1 # Out #2 # Name                 # Arguments
 # Mode # Units In # Units Out # Ratio  # Time (ms) #

 ######################################################################################################
   # 0  # 1      # -      # DFESolutionInjection # solutions=[]
 # -    # 0        # 1         # 0.00   # 0.01      #
   #    #        #        #                     # outSchema=[?1]
 #      #          #           #        #           #
```

```
#################################################################################################
   # 1  # 2      # 3      # DFETee               # -
# -    # 1        # 2        # 2.00   # 0.01      #

#################################################################################################
   # 2  # 4      # -      # DFEDistinctColumn    # column=?1
# -    # 1        # 1        # 1.00   # 0.21      #
   #    #        #        #                     # ordered=false
#      #          #          #        #           #

#################################################################################################
   # 3  # 5      # -      # DFEHashIndexBuild    # vars=[?1]
# -    # 1        # 1        # 1.00   # 0.03      #

#################################################################################################
   # 4  # 5      # -      # DFEPipelineJoin      # pattern=Edge((?1)-[?7:?5]->(?6))
# -    # 1        # 242      # 242.00 # 0.51      #
   #    #        #        #                     # constraints=[]
#      #          #          #        #           #
   #    #        #        #                     # patternEstimate=9223372036854775807
#      #          #          #        #           #

#################################################################################################
   # 5  # 6      # 7      # DFESync              # -
# -    # 243      # 243      # 1.00   # 0.02      #

#################################################################################################
   # 6  # 8      # -      # DFEForwardValue      # -
# -    # 1        # 1        # 1.00   # 0.01      #

#################################################################################################
   # 7  # 8      # -      # DFEForwardValue      # -
# -    # 242      # 242      # 1.00   # 0.02      #

#################################################################################################
   # 8  # 9      # -      # DFEHashIndexJoin     # -
# -    # 243      # 242      # 1.00   # 0.31      #

#################################################################################################
   # 9  # -      # -      # DFEDrain             # -
# -    # 242      # 0        # 0.00   # 0.01      #

#################################################################################################
```

```
   Runtime (ms)
   ============
   Query Execution: 11.744

   Traversal Metrics
   =================
   Step                                                               Count
Traversers       Time (ms)    % Dur

-------------------------------------------------------------------------------------------------
   DFEStep(Vertex)                                                      242
242          10.849    95.48
   NeptuneTraverserConverterDFEStep                                     242
242           0.514     4.52
                                                >TOTAL                    -
-          11.363        -

   Predicates
   ==========
   # of predicates: 18

   Results
   =======
   Count: 242


   Index Operations
   ================
   Query execution:
       # of statement index ops: 0
       # of terms materialized: 0
```

> **Note**
>
> La prise en charge du format DFE pour G705 étant une fonctionnalité expérimentale, le format exact de la `profile` sortie est sujet à modification.

Réglage des requêtes Gremlin à l'aide de la fonction Explique et du profil

Vous pouvez souvent ajuster les requêtes Gremlin dans Amazon Neptune pour améliorer les performances, en utilisant les informations mises à votre disposition dans les rapports que vous recevez des API explain et profile de Neptune. Pour ce faire, il est utile de comprendre comment Neptune traite les traversées Gremlin.

Important

Une modification a été apportée à TinkerPop la version 3.4.11 qui améliore l'exactitude du traitement des requêtes, mais qui, pour le moment, peut parfois avoir un impact sérieux sur les performances des requêtes.
Par exemple, une requête de ce type peut être beaucoup plus lente :

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  out()
```

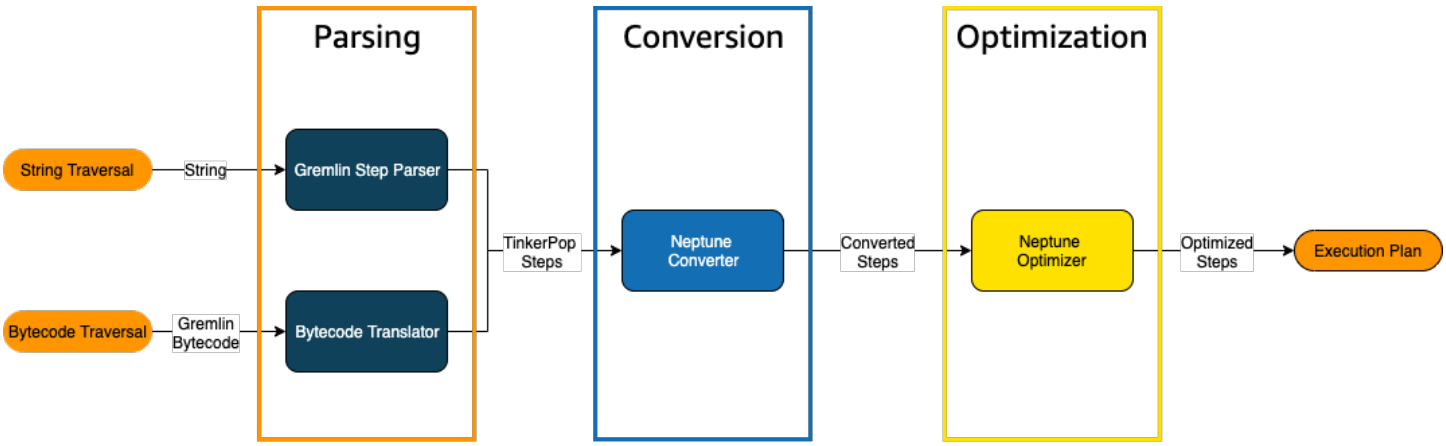
Les sommets après le pas de limite sont désormais récupérés de manière non optimale en raison de la modification 3.4.11. TinkerPop Pour éviter cela, vous pouvez modifier la requête en ajoutant l'étape barrier() à tout moment après `order().by()`. Par exemple :

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

TinkerPop La version 3.4.11 a été activée dans la version 1.0.5.0 du moteur Neptune.

Comprendre le traitement des traversées Gremlin dans Neptune

Lorsqu'une traversée Gremlin est envoyée à Neptune, trois processus principaux la convertissent en un plan d'exécution sous-jacent à exécuter par le moteur. Il s'agit de l'analyse, de la conversion et de l'optimisation :



Processus d'analyse des traversées

La première étape du traitement d'une traversée consiste à l'analyser dans un langage commun. Dans Neptune, ce langage commun est l'ensemble des TinkerPop étapes qui font partie de l'TinkerPopAPI. Chacune de ces étapes représente une unité de calcul au sein de la traversée.

Vous pouvez envoyer une traversée Gremlin à Neptune sous forme de chaîne ou de bytecode. Le point de terminaison REST et la méthode `submit()` du pilote client Java envoient des traversées sous forme de chaînes, comme dans cet exemple :

```
client.submit("g.V()")
```

Les applications et les pilotes de langage utilisant les variantes du langage Gremlin (GLV) envoient les traversées sous forme de bytecode.

Processus de conversion des traversées

La deuxième étape du traitement d'une traversée consiste à convertir ses TinkerPop étapes en un ensemble d'étapes Neptune converties et non converties. La plupart des étapes du langage de requête TinkerPop Apache G705 sont converties en Neptune-specific étapes optimisées pour être exécutées sur le moteur Neptune sous-jacent. Lorsqu'une TinkerPop étape sans équivalent Neptune est rencontrée dans une traversée, cette étape et toutes les étapes suivantes de la traversée sont traitées par le moteur de requête. TinkerPop

Pour plus d'informations sur les étapes qui peuvent être converties et dans quelles circonstances, consultez Prise en charge des étapes Gremlin.

Processus d'optimisation des traversées

La dernière étape du traitement des traversées consiste à exécuter la série d'étapes converties et non converties via l'optimiseur, afin de déterminer le meilleur plan d'exécution. Le résultat de cette optimisation est le plan d'exécution traité par le moteur Neptune.

Utilisation de l'API d'**`explication`** de Neptune Gremlin pour régler les requêtes

L'API Neptune explain n'est pas identique à l'étape Gremlin `explain()`. Elle renvoie le plan d'exécution final que le moteur Neptune peut traiter lors de l'exécution de la requête. Comme elle n'effectue aucun traitement, elle renvoie le même plan quels que soient les paramètres utilisés, et sa sortie ne contient aucune statistique sur l'exécution réelle.

Prenons l'exemple de la traversée simple suivante qui permet de trouver tous les sommets de l'aéroport pour Anchorage :

```
g.V().has('code','ANC')
```

Il existe deux façons d'exécuter cette traversée via l'API Neptune `explain`. La première méthode consiste à effectuer un appel REST au point de terminaison explain, comme suit :

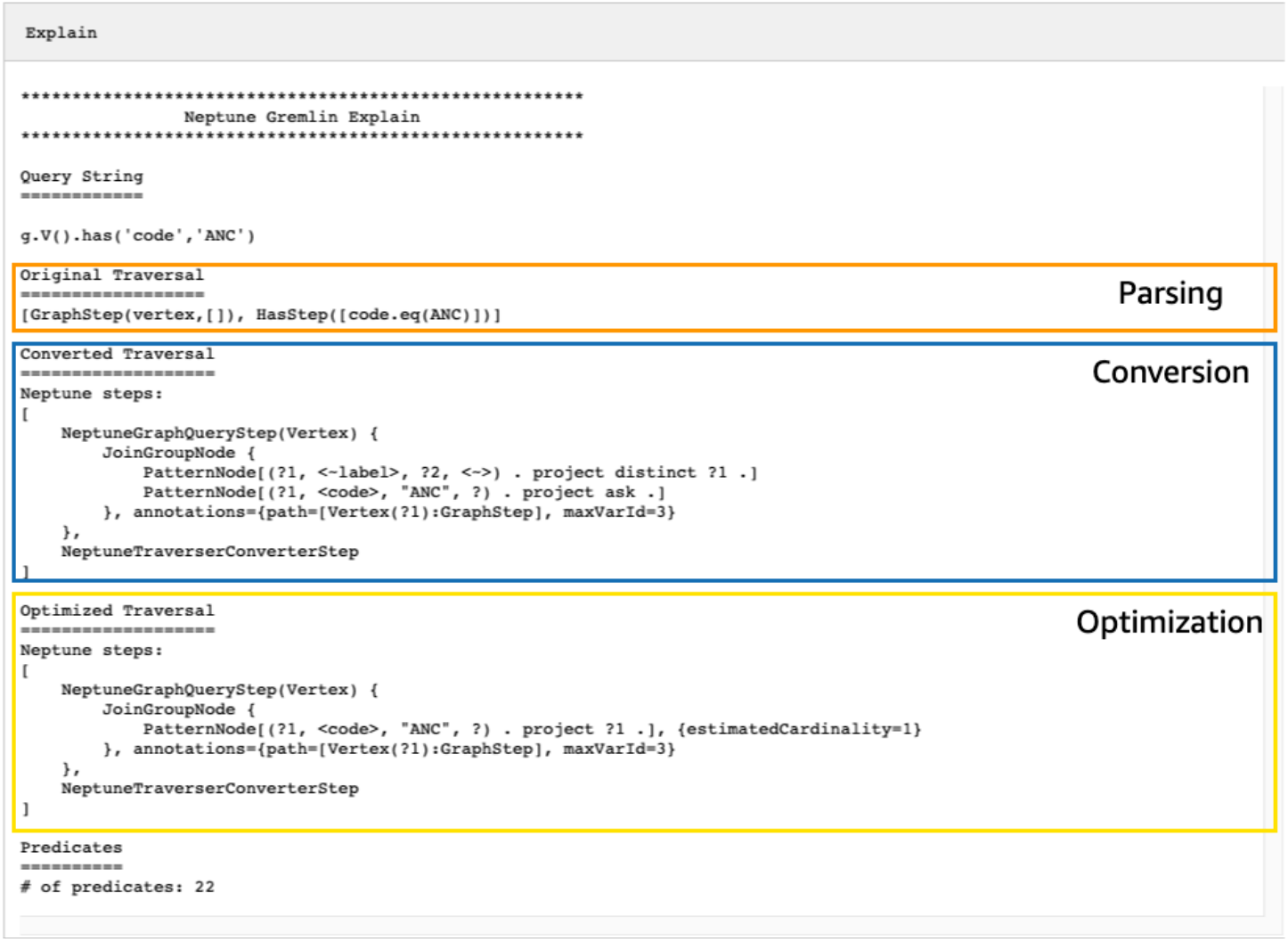
```
curl -X POST https://your-neptune-endpoint:port/gremlin/explain -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

La deuxième méthode consiste à utiliser la magie cellulaire %%gremlin du workbench Neptune avec le paramètre `explain`. Cela transmet le parcours contenu dans le corps de la cellule à l'API Neptune `explain`, puis affiche le résultat obtenu lorsque vous exécutez la cellule :

```
%%gremlin explain

g.V().has('code','ANC')
```

La sortie d'API `explain` qui en résulte décrit le plan d'exécution de Neptune pour la traversée. Comme vous pouvez le voir sur l'image ci-dessous, le plan inclut chacune des trois étapes du pipeline de traitement :



Réglage d'une traversée en examinant les étapes qui ne sont pas converties

L'une des premières choses à rechercher dans la sortie de l'API Neptune `explain` concerne les étapes Gremlin qui ne sont pas converties en étapes natives Neptune. Dans un plan de requête, lorsqu'une étape ne peut pas être convertie en étape native Neptune, elle est traitée par le serveur Gremlin, ainsi que toutes les étapes suivantes du plan.

Dans l'exemple ci-dessus, toutes les étapes de la traversée ont été converties. Examinons la sortie d'API `explain` pour cette traversée :

```
g.V().has('code','ANC').out().choose(hasLabel('airport'), values('code'), constant('Not
 an airport'))
```

Comme vous pouvez le voir sur l'image ci-dessous, Neptune n'a pas pu convertir l'étape `choose()` :

```
Explain
```

```
*******************************************************
               Neptune Gremlin Explain
*******************************************************

Query String
============

g.V().has('code','ANC').out().choose(hasLabel('airport'), values('code'), constant('Not an airport'))

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(OUT,vertex), ChooseStep([HasStep([~label.eq(airport)]), HasNex

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [ChooseStep([HasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [PropertiesSte

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .], {estimatedCardinality=1}
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .], {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [ChooseStep([NeptuneHasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [Proper

WARNING: >> ChooseStep([NeptuneHasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [PropertiesStep([code],value), E

Predicates
==========
# of predicates: 26
```

Il existe plusieurs façons d'ajuster les performances de la traversée. La première consiste à la réécrire de manière à éliminer l'étape qui n'a pas pu être convertie. Une autre solution consiste à déplacer l'étape vers la fin de la traversée afin que toutes les autres étapes puissent être converties en étapes natives.

Un plan de requête dont les étapes ne sont pas converties n'a pas toujours besoin d'être ajusté. Si les étapes qui ne peuvent pas être converties se situent à la fin de la traversée et sont liées à la mise en forme de la sortie plutôt qu'à la manière dont le graphe est parcouru, elles peuvent avoir peu d'effet sur les performances.

Lorsque vous examinez la sortie de l'API Neptune `explain`, tenez aussi compte des étapes qui n'utilisent pas d'index. La traversée suivante permet de trouver tous les aéroports dont les vols atterrissent à Anchorage :

```
g.V().has('code','ANC').in().values('code')
```

La sortie de l'API explain pour cette traversée est la suivante :

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============

g.V().has('code','ANC').in().values('code')

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(IN,vertex),
 PropertiesStep([code],value)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
            PatternNode[(?3, ?7, ?8, <~>) . project ?3,?8 . ContainsFilter(?7 in
 (<code>)) .]
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
```

```
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .],
 {estimatedCardinality=1}
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .],
 {estimatedCardinality=INFINITY}
            PatternNode[(?3, ?7=<code>, ?8, <~>) . project ?3,?8 .],
 {estimatedCardinality=7564}
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 26

WARNING: reverse traversal with no edge label(s) - .in() / .both() may impact query
 performance
```

Le message `WARNING` au bas de la sortie apparaît parce que l'étape `in()` de la traversée ne peut pas être gérée à l'aide de l'un des trois index gérés par Neptune (voir Comment les instructions sont indexées dans Neptune et Déclarations Gremlin dans Neptune). Comme l'étape `in()` ne contient aucun filtre d'arête, elle ne peut pas être résolue à l'aide de l'index `SPOG`, `POGS` ou `GPSO`. À la place, Neptune doit effectuer une analyse d'union pour trouver les sommets demandés, ce qui est beaucoup moins efficace.

Dans cette situation, vous pouvez ajuster la traversée de deux façons. La première consiste à ajouter un ou plusieurs critères de filtrage à l'étape `in()` afin qu'une recherche indexée puisse être utilisée pour résoudre la requête. Concernant l'exemple ci-dessus, cela peut être :

```
g.V().has('code','ANC').in('route').values('code')
```

La sortie de l'API Neptune `explain` pour la traversée révisée ne contient plus le message `WARNING` :

```
*******************************************************
                Neptune Gremlin Explain
```

```
**********************************************************

Query String
============

g.V().has('code','ANC').in('route').values('code')

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(IN,[route],vertex),
 PropertiesStep([code],value)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .
 ContainsFilter(?5 in (<route>)) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
            PatternNode[(?3, ?7, ?8, <~>) . project ?3,?8 . ContainsFilter(?7 in
 (<code>)) .]
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .],
 {estimatedCardinality=1}
            PatternNode[(?3, ?5=<route>, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?
6) .], {estimatedCardinality=32042}
            PatternNode[(?3, ?7=<code>, ?8, <~>) . project ?3,?8 .],
 {estimatedCardinality=7564}
```

```
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 26
```

Si vous exécutez de nombreuses traversées de ce type, vous pouvez également les exécuter dans un cluster de bases de données Neptune dont l'index OSGP facultatif est activé (voir Activation d'un index OSGP). L'activation d'un index OSGP présente des inconvénients :

- Elle doit être activée dans un cluster de bases de données avant tout chargement de données.
- Les taux d'insertion des sommets et des arêtes peuvent ralentir de jusqu'à 23 %.
- L'utilisation du stockage augmente d'environ 20 %.
- Les requêtes de lecture qui répartissent les demandes sur tous les index peuvent accroître les temps de latence.

L'utilisation d'un index OSGP est tout à fait justifiée pour un ensemble restreint de modèles de requêtes, mais à moins que vous ne les exécutiez fréquemment, il est généralement préférable de faire en sorte que les traversées que vous écrivez puissent être résolues à l'aide des trois index principaux.

Utilisation d'un grand nombre de prédicats

Neptune traite chaque étiquette d'arête et chaque nom de propriété de sommet ou d'arête distinct dans le graphe comme un prédicat. Il est conçu par défaut pour fonctionner avec un nombre relativement faible de prédicats distincts. Lorsque les données du graphe contiennent plus de quelques milliers de prédicats, les performances peuvent se dégrader.

La sortie Neptune `explain` vous en avertit si c'est le cas :

```
Predicates
==========
# of predicates: 9549
WARNING: high predicate count (# of distinct property names and edge labels)
```

S'il n'est pas pratique de retravailler votre modèle de données pour réduire le nombre d'étiquettes et de propriétés, et donc le nombre de prédicats, le meilleur moyen d'ajuster les traversées consiste à les exécuter dans un cluster de bases de données dont l'index OSGP est activé, comme indiqué ci-dessus.

Utilisation de l'API de **`profil`** Neptune Gremlin pour ajuster les traversées

L'API Neptune `profile` est très différente de l'étape Gremlin `profile()`. Comme l'API `explain`, sa sortie inclut le plan de requête que le moteur Neptune utilisera lors de l'exécution de la traversée. En outre, la sortie `profile` inclut les statistiques d'exécution réelles de la traversée, compte tenu de la façon dont ses paramètres sont définis.

Là aussi, prenons l'exemple de la traversée simple qui permet de trouver tous les sommets de l'aéroport pour Anchorage :

```
g.V().has('code','ANC')
```

Comme avec l'API `explain`, vous pouvez invoquer l'API `profile` à l'aide d'un appel REST :

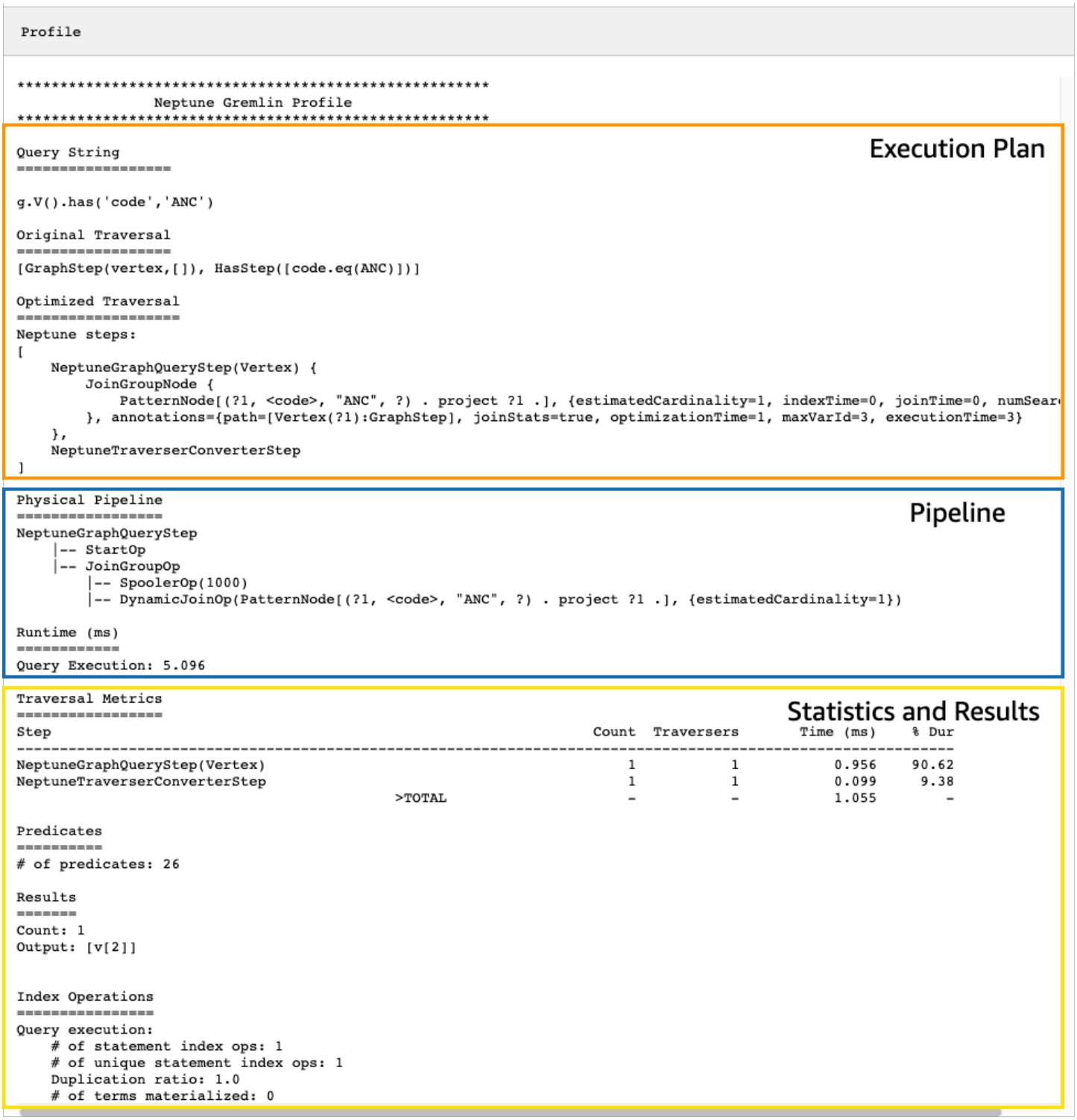
```
curl -X POST https://your-neptune-endpoint:port/gremlin/profile -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

Utilisez également la magie cellulaire %%gremlin du workbench Neptune avec le paramètre `profile`. Cela transmet le parcours contenu dans le corps de la cellule à l'API Neptune `profile`, puis affiche le résultat obtenu lorsque vous exécutez la cellule :

```
%%gremlin profile

g.V().has('code','ANC')
```

La sortie d'API `profile` qui en résulte contient à la fois le plan d'exécution de Neptune pour la traversée et les statistiques relatives à l'exécution du plan, comme vous pouvez le voir sur cette image :



Dans la sortie `profile`, la section du plan d'exécution contient uniquement le plan d'exécution final de la traversée, et non les étapes intermédiaires. La section du pipeline contient les opérations physiques du pipeline qui ont été effectuées ainsi que le temps réel (en millisecondes) nécessaire à

l'exécution de la traversée. La métrique d'exécution est extrêmement utile pour comparer le temps nécessaire à deux versions différentes d'une traversée lorsque vous les optimisez.

> **Note**
>
> La durée d'exécution initiale d'une traversée est généralement plus longue que celle des exécutions suivantes, car la première entraîne la mise en cache des données pertinentes.

La troisième section de la sortie `profile` contient les statistiques d'exécution et les résultats de la traversée. Pour voir comment ces informations peuvent être utiles pour ajuster une traversée, prenons l'exemple de la traversée suivante, qui permet de trouver tous les aéroports dont le nom commence par « Anchora » et tous les aéroports accessibles en deux étapes à partir de ces aéroports, en renvoyant le code des aéroports, les itinéraires de vol et les distances :

```
%%gremlin profile

g.withSideEffect("Neptune#fts.endpoint", "{your-OpenSearch-endpoint-URL").
    V().has("city", "Neptune#fts Anchora~").
    repeat(outE('route').inV().simplePath()).times(2).
    project('Destination', 'Route').
        by('code').
        by(path().by('code').by('dist'))
```

Mesures de traversée dans la sortie de l'API de **`profil Neptune`**

Le premier ensemble de métriques disponible dans toutes les sorties `profile` est celui des métriques de traversée. Elles sont similaires aux métriques de l'étape Gremlin `profile()`, à quelques différences près :

```
Traversal Metrics
=================
Step                                                               Count  Traversers
    Time (ms)    % Dur
-------------------------------------------------------------------------------------------------
NeptuneGraphQueryStep(Vertex)                                       3856        3856
       91.701     9.09
NeptuneTraverserConverterStep                                       3856        3856
       38.787     3.84
ProjectStep([Destination, Route],[value(code), ...                  3856        3856
      878.786    87.07
```

```
  PathStep([value(code), value(dist)])                                3856        3856
     601.359
                                                >TOTAL                    -           -
    1009.274        -
```

La première colonne du tableau des métriques de traversée répertorie les étapes exécutées par la traversée. Les deux premières étapes sont généralement les Neptune-specific étapes, `NeptuneGraphQueryStep` et`NeptuneTraverserConverterStep`.

`NeptuneGraphQueryStep` représente le temps d'exécution pour toute la partie de la traversée qui pourrait être convertie et exécutée nativement par le moteur Neptune.

`NeptuneTraverserConverterStep`représente le processus de conversion de la sortie de ces étapes converties en TinkerPop traverseurs qui permettent de traiter les étapes qui n'ont pas pu être converties, le cas échéant, ou de renvoyer les résultats dans un TinkerPop-compatible format.

Dans l'exemple ci-dessus, nous avons plusieurs étapes non converties. Nous voyons donc que chacune de ces TinkerPop étapes (`ProjectStep`,`PathStep`) apparaît alors sous forme de ligne dans le tableau.

La deuxième colonne du tableau indique le nombre de traversers représentés qui sont passés par l'étape, tandis que la troisième colonne indique le nombre de traverseurs qui sont passés par cette étape, comme expliqué dans la documentation de l'étape du TinkerPop profil. `Count` `Traversers`

Dans notre exemple, 3 856 sommets et 3 856 traverseurs sont renvoyés par `NeptuneGraphQueryStep`, et ces nombres restent les mêmes pendant le reste du traitement car `ProjectStep` et `PathStep` mettent en forme les résultats, sans les filtrer.

> **Note**
>
> Contrairement à cela TinkerPop, le moteur Neptune n'optimise pas les performances en augmentant le nombre de ses étapes et de ses `NeptuneGraphQueryStep` étapes. `NeptuneTraverserConverterStep` Le groupage est l' TinkerPopopération qui combine des traverseurs situés sur le même sommet afin de réduire la surcharge opérationnelle, et c'est ce qui explique la différence entre les `Traversers` nombres `Count` et. Comme le groupage ne se produit que par étapes déléguées TinkerPop par Neptune, et non par étapes gérées nativement par Neptune, `Count` les colonnes et diffèrent rarement. `Traverser`

La colonne Temps indique le nombre de millisecondes que l'étape a duré, tandis que la colonne `% Dur` indique le pourcentage du temps de traitement total que l'étape a pris. Ces métriques vous indiquent sur quoi concentrer vos efforts de réglage en présentant les étapes qui ont pris le plus de temps.

Indicer les mesures des opérations dans la sortie de l'API de **`profil`** Neptune

Les opérations d'indexation constituent un autre ensemble de métriques figurant dans les résultats de l'API de profil Neptune :

```
Index Operations
================
Query execution:
    # of statement index ops: 23191
    # of unique statement index ops: 5960
    Duplication ratio: 3.89
    # of terms materialized: 0
```

Ces métriques fournissent les informatons suivantes :

- Nombre total de recherches d'index.
- Nombre de recherches d'index uniques effectuées.
- Ratio entre le nombre total de recherches d'index et le nombre total de recherches uniques. Un ratio inférieur indique une redondance moindre.
- Nombre de termes matérialisés à partir du dictionnaire.

Répéter les métriques dans la sortie de l'API de **`profil`** Neptune

Si votre traversée utilise une étape `repeat()` comme dans l'exemple ci-dessus, une section contenant des métriques de répétition apparaît dans la sortie `profile` :

```
Repeat Metrics
==============
Iteration  Visited   Output    Until     Emit     Next
-------------------------------------------------------
        0        2        0        0        0        2
        1       53        0        0        0       53
        2     3856     3856     3856        0        0
-------------------------------------------------------
```

```
              3911      3856      3856         0        55
```

Ces métriques fournissent les informatons suivantes :

- Nombre de boucles pour une ligne (colonne `Iteration`)
- Nombre d'éléments auxquels la boucle a accédé (colonne `Visited`)
- Nombre d'éléments générés par la boucle (colonne `Output`)
- Dernier élément généré par la boucle (colonne `Until`)
- Nombre d'éléments émis par la boucle (colonne `Emit`)
- Nombre d'éléments étant passés de la boucle à la boucle suivante (colonne `Next`)

Ces métriques de répétition sont très utiles pour comprendre le facteur de ramification de la traversée. Elles vous permettent de vous faire une idée de la quantité de travail effectuée par la base de données. Vous pouvez utiliser ces chiffres pour diagnostiquer les problèmes de performances, en particulier lorsqu'une même traversée fonctionne de manière radicalement différente selon les paramètres.

Full-text métriques de recherche dans la sortie de l'API de **`profil`** Neptune

Lorsqu'une traversée utilise une recherche en texte intégral, comme dans l'exemple ci-dessus, une section contenant les métriques de recherche en texte intégral (FTS) apparaît dans la sortie `profile` :

```
FTS Metrics
==============
SearchNode[(idVar=?1, query=Anchora~, field=city) . project ?1 .],
    {endpoint=your-OpenSearch-endpoint-URL, incomingSolutionsThreshold=1000,
 estimatedCardinality=INFINITY,
    remoteCallTimeSummary=[total=65, avg=32.500000, max=37, min=28],
    remoteCallTime=65, remoteCalls=2, joinTime=0, indexTime=0, remoteResults=2}

    2 result(s) produced from SearchNode above
```

Cela montre la requête envoyée au cluster ElasticSearch (ES) et indique plusieurs mesures relatives à l'interaction avec ElasticSearch lesquelles vous pouvez identifier les problèmes de performances liés à la recherche en texte intégral :

- Informations récapitulatives sur les appels dans l' ElasticSearch index :

  - Nombre total de millisecondes requis par tous les appels à distance pour satisfaire la requête (`total`).
  - Nombre moyen de millisecondes passées dans un appel à distance (`avg`).
  - Nombre minimum de millisecondes passées dans un appel à distance (`min`).
  - Nombre maximum de millisecondes passées dans un appel à distance (`max`).
- Durée totale consommée par RemoteCalls to ElasticSearch ()`remoteCallTime`.
- Le nombre d'appels à distance effectués vers ElasticSearch ()`remoteCalls`.
- Le nombre de millisecondes passées à joindre les ElasticSearch résultats (). `joinTime`
- Nombre de millisecondes passées dans les recherches d'index (`indexTime`).
- Le nombre total de résultats renvoyés par ElasticSearch (`remoteResults`).

Prise en charge des étapes Gremlin natives dans Amazon Neptune

Le moteur Amazon Neptune n'offre actuellement pas de prise en charge native complète pour toutes les étapes Gremlin, comme expliqué dans Réglage des requêtes Gremlin. La prise en charge actuelle se divise en quatre catégories :

- Étapes Gremlin qui peuvent toujours être converties en opérations natives du moteur Neptune
- Étapes Gremlin qui peuvent être converties en opérations natives du moteur Neptune dans certains cas
- Étapes Gremlin qui sont jamais converties en opérations natives du moteur Neptune
- Étapes Gremlin qui ne sont pas du tout prises en charge dans Neptune

Étapes Gremlin qui peuvent toujours être converties en opérations natives du moteur Neptune

De nombreuses étapes Gremlin peuvent être converties en opérations natives du moteur Neptune tant qu'elles répondent aux conditions suivantes :

- Elles ne sont pas précédées dans la requête d'une étape qui ne peut pas être convertie.
- Leur étape parent, le cas échéant, peut être convertie.
- Toutes leurs traversées enfants, le cas échéant, peuvent être converties.

Les étapes Gremlin suivantes sont toujours converties en opérations natives du moteur Neptune si elles répondent à ces conditions :

- and( )
- as( )
- count( )
- E( )
- emit( )
- explain( )
- group( )
- groupCount( )
- identity( )
- is( )
- key( )
- label( )
- limit( )
- local( )
- loops( )
- not( )
- or( )
- profile( )
- properties( )
- subgraph( )
- until( )
- V( )
- value( )
- valueMap( )
- values( )

Étapes Gremlin qui peuvent être converties en opérations natives du moteur Neptune dans certains cas

Certaines étapes Gremlin peuvent être converties en opérations natives du moteur Neptune dans certaines situations, mais pas dans d'autres :

- addE( ) : l'étape `addE()` peut généralement être convertie en une opération native du moteur Neptune, à moins qu'elle ne soit immédiatement suivie d'une étape `property()` contenant une traversée en tant que clé.
- addV( ) : l'étape `addV()` peut généralement être convertie en une opération native du moteur Neptune, à moins qu'elle ne soit immédiatement suivie d'une étape `property()` contenant une traversée en tant que clé ou à moins que plusieurs étiquettes ne soient attribuées.
- aggregate( ) : l'étape `aggregate()` peut généralement être convertie en une opération native du moteur Neptune, sauf si elle est utilisée dans une traversée ou une sous-traversée secondaire, ou à moins que la valeur stockée ne soit autre qu'un sommet, une arête, un ID, une étiquette ou une valeur de propriété.

  Dans l'exemple ci-dessous, l'étape `aggregate()` n'est pas convertie, car elle est utilisée dans une traversée enfant :

  ```
  g.V().has('code','ANC').as('a')
       .project('flights').by(select('a')
       .outE().aggregate('x'))
  ```

  Dans cet exemple, l'étape aggregate() n'est pas convertie, car ce qui est stocké est une valeur `min()` :

  ```
  g.V().has('code','ANC').outE().aggregate('x').by(values('dist').min())
  ```

- barrier( ) : l'étape `barrier()` peut généralement être convertie en une opération native du moteur Neptune, sauf si l'étape qui la suit n'est pas convertie.
- cap( ) : le seul cas où l'étape `cap()` est convertie est lorsqu'elle est combinée à l'étape `unfold()` pour renvoyer une version dépliée d'un agrégat de valeurs de sommet, d'arête, d'ID ou de propriété. Dans cet exemple, `cap()` sera converti, car il est suivi de `.unfold()` :

  ```
  g.V().has('airport','country','IE').aggregate('airport').limit(2)
       .cap('airport').unfold()
  ```

  Toutefois, si vous supprimez `.unfold()`, `cap()` ne sera pas converti :

  ```
  g.V().has('airport','country','IE').aggregate('airport').limit(2)
       .cap('airport')
  ```

- coalesce () — Le seul cas où l'`coalesce()`étape est convertie est lorsqu'elle suit le modèle Upsert recommandé sur la TinkerPop page des recettes. Les autres modèles coalesce() ne sont pas autorisés. La conversion est limitée au cas où toutes les traversées enfants peuvent être converties, où elles génèremt toutes le même type de sortie (sommet, arête, ID, valeur, clé ou étiquette), où elles mènent toutes à un nouvel élément et où elles ne contiennent pas l'étape `repeat()`.
- constant( ) : l'étape constant() n'est actuellement convertie que si elle est utilisée dans une partie `sack().by()` d'une traversée pour attribuer une valeur de constante, comme ceci :

```
g.V().has('code','ANC').sack(assign).by(constant(10)).out().limit(2)
```

- cyclicPath( ) : l'étape `cyclicPath()` peut généralement être convertie en opération native du moteur Neptune, sauf si l'étape est utilisée avec des modulateurs `by()`, `from()` ou `to()`. Dans les requêtes suivantes, par exemple, `cyclicPath()` n'est pas converti :

```
g.V().has('code','ANC').as('a').out().out().cyclicPath().by('code')
g.V().has('code','ANC').as('a').out().out().cyclicPath().from('a')
g.V().has('code','ANC').as('a').out().out().cyclicPath().to('a')
```

- drop( ) : l'étape `drop()` peut généralement être convertie en opération native du moteur Neptune, sauf si l'étape est utilisée dans une étape `sideEffect()` ou `optional()`.
- fold () — L'étape fold () ne peut être convertie que dans deux situations, à savoir lorsqu'elle est utilisée selon le modèle Upsert recommandé sur la page des TinkerPop recettes et lorsqu'elle est utilisée dans un `group().by()` contexte comme celui-ci :

```
g.V().has('code','ANC').out().group().by().by(values('code', 'city').fold())
```

- has () — L'étape `has () `peut généralement être convertie en une opération native du moteur Neptune à condition que les requêtes avec « T » utilisent le prédicat` `, P.eqP.neq` ou ``. P.contains Attendez-vous à des variations de `has () `qui impliquent que les instances de `P` soient également converties en mode natif, comme `hasId ('id1234')` qui est équivalent à `has (eq, 'id1234') `. T.id
- id( ) : l'étape `id()` est convertie sauf si elle est utilisée au niveau d'une propriété, comme ceci :

```
g.V().has('code','ANC').properties('code').id()
```

- MerGee () — L'`mergeE()`étape peut être convertie en une opération native du moteur Neptune si les paramètres (la condition de fusion, on`Create` le on`Match` et) sont constants (`null`soit

une constante, soit une Map constante). `select()` `Map` Tous les exemples d'arêtes ascendantes peuvent être convertis.

- mergeV () — L'étape mergeV () peut être convertie en un fonctionnement natif du moteur Neptune si les paramètres (la condition de fusion, on`Create` etonMatch) sont constants (soit une constante, soit une constante`null`, soit une valeur de a). `Map` `select()` `Map` Tous les exemples de sommets ascendants peuvent être convertis.
- order( ) : l'étape `order()` peut généralement être convertie en opération native du moteur Neptune, sauf si l'une des conditions suivantes est vraie :
  - L'étape `order()` se trouve dans une traversée enfant imbriquée, comme ceci :

    ```
    g.V().has('code','ANC').where(V().out().order().by(id))
    ```

  - L'ordre local est utilisé (par exemple, avec `order(local)`).
  - Un comparateur personnalisé est utilisé dans la modulation `by()` pour effectuer le tri. Voici un exemple de l'utilisation de `sack()` :

    ```
    g.withSack(0).
      V().has('code','ANC').
          repeat(outE().sack(sum).by('dist').inV()).times(2).limit(10).
          order().by(sack())
    ```

  - Il existe plusieurs ordres pour le même élément.
- project( ) : l'étape `project()` peut généralement être convertie en opération native du moteur Neptune, sauf si le nombre d'instructions `by()` qui suivent `project()` ne correspond pas au nombre d'étiquettes spécifié, comme ici :

  ```
  g.V().has('code','ANC').project('x', 'y').by(id)
  ```

- range( ) : l'étape `range()` n'est convertie que lorsque la limite inférieure de la plage en question est égale à zéro (par exemple, `range(0,3)`).
- repeat( ) : l'étape `repeat()` peut généralement être convertie en opération native du moteur Neptune, sauf si elle est imbriquée dans une autre étape `repeat()`, comme ceci :

  ```
  g.V().has('code','ANC').repeat(out().repeat(out()).times(2)).times(2)
  ```

- sack( ) : l'étape `sack()` peut généralement être convertie en opération native du moteur Neptune, sauf dans les cas suivants :

  - Si un opérateur sack non numérique est utilisé.
  - Si un opérateur sack non numérique autre que `+`, `-`, `mult`, `div`, `min` et `max` est utilisé.
  - Si `sack()` est utilisé dans une étape `where()` pour filtrer les données en fonction d'une valeur sack, comme ici :

```
g.V().has('code','ANC').sack(assign).by(values('code')).where(sack().is('ANC'))
```

- sum( ) : l'étape `sum()` peut généralement être convertie en opération native du moteur Neptune, mais pas lorsqu'elle est utilisée pour calculer une somme globale, comme ceci :

```
g.V().has('code','ANC').outE('routes').values('dist').sum()
```

- union( ) : l'étape `union()` peut être convertie en opération native du moteur Neptune tant qu'il s'agit de la dernière étape de la requête en dehors de l'étape terminale.
- unfold () — L'`unfold()`étape ne peut être convertie en une opération native du moteur Neptune que lorsqu'elle est utilisée selon le modèle Upsert recommandé sur TinkerPopla page des recettes, et lorsqu'elle est utilisée conjointement avec `cap()` ceci :

```
g.V().has('airport','country','IE').aggregate('airport').limit(2)
     .cap('airport').unfold()
```

- where( ) : l'étape `where()` peut généralement être convertie en opération native du moteur Neptune, sauf dans les cas suivants :
  - Lorsque des modulations by() sont utilisées, comme ceci :

```
g.V().hasLabel('airport').as('a')
     .where(gt('a')).by('runways')
```

  - Lorsque des opérateurs de comparaison autres que `eq`, `neq`, `within` et `without` sont utilisés.
  - Lorsque des agrégations fournies par l'utilisateur sont utilisées.

Étapes Gremlin qui sont jamais converties en opérations natives du moteur Neptune

Les étapes Gremlin suivantes sont prises en charge dans Neptune, mais ne sont jamais converties en opérations natives du moteur Neptune. Au lieu de cela, elles sont exécutées par le serveur Gremlin.

- choose( )

- coin( )
- échec ()
- inject( )
- match( )
- math( )
- max( )
- mean( )
- min( )
- option( )
- optional( )
- path( )
- propertyMap( )
- sample( )
- skip( )
- tail( )
- timeLimit( )
- tree( )

Étapes Gremlin qui ne sont pas du tout prises en charge dans Neptune

Les étapes Gremlin suivantes ne sont pas du tout prises en charge dans Neptune. Dans la plupart des cas, cela est dû au fait qu'elles nécessitent un `GraphComputer`, ce que Neptune ne prend pas en charge actuellement.

- connectedComponent( )
- io( )
- shortestPath( )
- withComputer( )
- pageRank( )
- peerPressure( )
- program( )

L'étape `io()` est partiellement prise en charge, dans la mesure où elle peut être utilisée pour effectuer une opération de type `read()` à partir d'une URL, mais pas une opération de type `write()`.

Utilisation de Gremlin avec le moteur de requêtes Neptune DFE

Vous pouvez activer le moteur de requête alternatif de Neptune, connu sous le nom de DFE, de deux manières :

- Définissez le paramètre d'`neptune_dfe_query_engine`instance sur `enabled` pour toutes les requêtes sur l'instance.
- Utilisez l'indice de `useDFE` requête sur des requêtes individuelles (`neptune_dfe_query_engine`doit être défini sur sa valeur par défaut `deviaQueryHint`).

Lorsque DFE est activé, Neptune traduit les requêtes et les traversées Gremlin en lecture seule en une représentation logique intermédiaire et les exécute sur le moteur DFE chaque fois que cela est possible.

Cependant, le DFE ne prend pas encore en charge toutes les étapes Gremlin. Lorsqu'une étape ne peut pas être exécutée en mode natif sur le DFE, Neptune recourt à nouveau TinkerPop pour exécuter l'étape. Les rapports `explain` et `profile` incluent des avertissements lorsque cela se produit.

Couverture des étapes de Garmlin en PDF

Gremlin DFE est une fonctionnalité expérimentale qui peut être utilisée soit en activant le paramètre d'instance, soit en utilisant l'indice de `Neptune#useDFE` requête. Pour plus d'informations, reportez-vous à la section Utilisation de Gkremlin avec le moteur de requêtes Neptune DFE.

Les étapes suivantes peuvent être utilisées dans Gremlin DFE.

Trajectoire et étapes de traversée :

asDate () , barrier (), call (), cap (), DateAdd (), DateDiff (), disjunct (), drop (), fail (), filter (), flatMap (), id (), identité (), index (), intersection (), injection (), label (), longueur (), boucles (), carte (), ordre (), ordre ()), chemin (), project (), range (), repeat (), reverse (), sack (), sample (), select (), sideEffect (), split (), unfold (), union ()

Étapes d'agrégation et de collecte :

agréger (global), combiner (), compter (), déduplication (), déduplication (local), pliage (), groupe (), groupCount (),

Étapes mathématiques :

max (), moyenne (), min (), somme ()

Étapes de l'élément :

OtherV (), ElementMap (), élément (), v (), out (), in (), les deux (), OutE (), iNE (), BothE (), OutV (), InV (), BothV (), AutreV ()

Étapes de la propriété :

propriétés (), clé (), valueMap (), valeur ()

Étapes du filtre :

et (), coalesce (), coin (), has (), is (), local (), none (), not (), ou (), où ()

Étapes de manipulation des chaînes :

concat () , LTrim (), RTrim (), sous-chaîne (), ToLower (), ToUpper (), trim ()

Prédicats :

- Comparez : eq, neq, lt, lte, gt, gte
- Contient : dedans, sans
- texTP : se terminant par, contenant, non, nonStartingWith, notContaining EndingWith
- P : et, ou, entre, à l'extérieur, à l'intérieur

Limitations

Répéter avec limite. Les libellés contenus dans la procédure de traversée répétée et de déduplication ne sont pas encore pris en charge dans DFE.

```
// With Limit inside the repeat traversal
  g.V().has('code','AGR').repeat(out().limit(5)).until(has('code','FRA'))

  // With Labels inside the repeat traversal
  g.V().has('code','AGR').repeat(out().as('a')).until(has('code','FRA'))

  // With Dedup inside the repeat traversal
  g.V().has('code','AGR').repeat(out().dedup()).until(has('code','FRA'))
```

Les chemins comportant des répétitions imbriquées ou des étapes de branchement ne sont pas encore pris en charge.

```
// Path with branching steps
  g.V().has('code','AGR').union(identity, outE().inV()).path().by('code')


  // With nested repeat
  g.V().has('code','AGR').repeat(out().union(identity(), out())).path().by('code')
```

Entrelacement de la planification des requêtes

Lorsque le processus de conversion identifie une étape Gremlin qui n'a pas d'opérateur DFE natif correspondant, avant de revenir à Tinkerpop, il essaie de trouver d'autres parties de requête intermédiaires pouvant être exécutées nativement sur le moteur DFE. Pour ce faire, il applique une logique d'entrelacement à la traversée de niveau supérieur. De la sorte, les étapes prises en charge sont utilisées dans la mesure du possible.

Toute conversion de requête intermédiaire sans préfixe est représentée à l'aide de `NeptuneInterleavingStep` dans les sorties `explain` et `profile`.

Pour comparer les performances, vous pouvez désactiver l'entrelacement dans une requête, tout en utilisant le moteur DFE pour exécuter la partie avec préfixe. Vous pouvez également utiliser uniquement le TinkerPop moteur pour l'exécution de requêtes sans préfixe. Pour ce faire, vous avez besoin d'un indicateur de requête `disableInterleaving`.

Tout comme l'indicateur de requête useDFE avec la valeur `false` empêche totalement l'exécution d'une requête sur le DFE, l'indicateur requête `disableInterleaving` avec la valeur `true` désactive l'entrelacement DFE pour la conversion d'une requête. Par exemple :

```
g.with('Neptune#disableInterleaving', true)
```

```
.V().has('genre','drama').in('likes')
```

Mise à jour de l'**`explication`** et de la sortie de **`profil`** de Gremlin

Gremlin explain fournit des informations sur la traversée optimisée que Neptune utilisera pour exécuter une requête. Consultez l'exemple de sortie DFE `explain` pour voir ce à quoi ressemble la sortie `explain` lorsque le moteur DFE est activé.

L'API Gremlin `profile` effectue une traversée Gremlin spécifiée, collecte diverses métriques relatives à l'exécution et génère un rapport de profil contenant des informations sur le plan de requête optimisé et les statistiques d'exécution de différents opérateurs. Consultez l'exemple de sortie DFE `profile` pour voir ce à quoi ressemble la sortie `profile` lorsque le moteur DFE est activé.

> **Note**
>
> La prise en charge du format DFE pour G705 étant une fonctionnalité expérimentale, le format exact de la `profile` sortie `explain` et est sujet à modification.

Accès au graphe Neptune avec openCypher

Neptune prend en charge la création d'applications de graphe à l'aide d'openCypher, qui est actuellement l'un des langages de requête les plus populaires pour les développeurs travaillant avec des bases de données orientées graphe. Les développeurs, les analystes commerciaux et les scientifiques des données apprécient la SQL-inspired syntaxe d'OpenCypher, car elle fournit une structure familière pour composer des requêtes pour les applications graphiques.

openCypher est un langage de requête déclaratif pour les graphes de propriétés initialement développé par Neo4j, puis rendu open source en 2015. Il a contribué au projet openCypher sous une licence open source Apache 2. Sa syntaxe est documentée dans Cypher Query Language Reference, Version 9 (Référence du langage de requête Cypher, version 9).

Pour connaître les limites et les différences de prise en charge de la spécification openCypher dans Neptune, consultez Conformité aux spécifications OpenCypher dans Amazon Neptune.

> **Note**
>
> L'implémentation actuelle du langage de requête Cypher dans Neo4j s'écarte à certains égards de la spécification openCypher. Si vous migrez le code Neo4j Cypher actuel vers

Neptune, consultez Compatibilité de Neptune avec Neo4j et Réécriture des requêtes Cypher pour les exécuter dans openCypher sur Neptune pour obtenir de l'aide.

À partir de la version 1.1.1.0 du moteur, openCypher est disponible pour une utilisation en production dans Neptune.

Comparaison de Gremlin et openCypher : similarités et différences

Gremlin et openCypher sont tous deux des langages de requête basés sur des graphes de propriétés. Ils sont complémentaires à bien des égards.

Gremlin est destiné aux programmeurs et s'intègre parfaitement dans le code. Par conséquent, Gremlin utilise une syntaxe impérative dès sa conception, tandis que la syntaxe déclarative d'openCypher peut sembler plus familière aux personnes ayant de l'expérience en SQL ou en SPARQL. Gremlin peut sembler plus naturel à un scientifique des données utilisant Python dans un bloc-notes Jupyter, tandis qu'openCypher peut sembler plus intuitif à un professionnel ayant une certaine expérience en SQL.

L'avantage est que vous n'avez pas à choisir entre Gremlin et openCypher dans Neptune. Les requêtes dans l'un ou l'autre de ces langages fonctionnent sur le même graphe, quelle que soit le langage utilisé pour entrer ces données. Vous trouverez peut-être plus pratique d'utiliser Gremlin pour certaines tâches et openCypher pour d'autres, selon ce que vous faites.

Gremlin utilise une syntaxe impérative qui vous permet de contrôler la façon dont vous vous déplacez dans le graphe en une série d'étapes, chacune prenant en charge un flux de données, effectuant une action associée (à l'aide d'un filtre, d'un mappage, etc.), puis transmettant les résultats à l'étape suivante. Une requête Gremlin prend généralement la forme `g.V()`, suivie d'étapes supplémentaires.

Dans openCypher, vous utilisez une syntaxe déclarative, inspirée de SQL, qui spécifie un modèle de nœuds et de relations à rechercher dans le graphe à l'aide d'une syntaxe de motif (comme `()-[]->()`). Une requête openCypher commence souvent par une clause `MATCH`, suivie d'autres clauses telles que `WHERE`, `WITH`, et `RETURN`

Premiers pas avec openCypher

Vous pouvez interroger les données du graphe de propriétés dans Neptune à l'aide d'openCypher, quelle que soit la manière dont elles ont été chargées, mais vous ne pouvez pas utiliser openCypher pour interroger des données chargées au format RDF.

Le chargeur en bloc Neptune accepte les données du graphe de propriétés au format CSV pour Gremlin et au format CSV pour openCypher. Bien entendu, vous pouvez également ajouter des données de propriété à votre graphique à l'aide des requêtes and/or OpenCypher de Gkremlin.

De nombreux didacticiels en ligne permettent d'apprendre le langage de requête Cypher. Voici quelques exemples rapides de requêtes openCypher qui vous aideront à vous familiariser avec ce langage, mais le moyen le plus efficace et le plus simple de commencer à utiliser openCypher pour interroger le graphe Neptune est d'utiliser les blocs-notes openCypher dans le workbench Neptune. Le workbench est open source et est hébergé sur GitHub . https://github.com/aws-samples/amazon-neptune-samples

Vous trouverez les blocs-notes OpenCypher dans le référentiel Neptune GitHub graph-notebook. Consultez en particulier la Air-routesvisualisation et les carnets Premier Teams en anglais pour OpenCypher.

Les données traitées par OpenCypher prennent la forme d'une série non ordonnée de cartes. key/value Le principal moyen d'affiner, de manipuler et d'augmenter ces cartes consiste à utiliser des clauses qui exécutent des tâches telles que la correspondance de modèles, l'insertion, la mise à jour et la suppression sur les key/value paires.

openCypher contient plusieurs clauses permettant de trouver des modèles de données dans le graphe. `MATCH` est la plus courante. `MATCH` vous permet de spécifier le modèle de nœuds, de relations et de filtres que vous souhaitez rechercher dans le graphe. Par exemple :

- Obtenir tous les nœuds

```
MATCH (n) RETURN n
```

- Trouver les nœuds connectés

```
MATCH (n)-[r]->(d) RETURN n, r, d
```

- Trouver un chemin

```
MATCH p=(n)-[r]->(d) RETURN p
```

- Obtenir tous les nœuds avec une étiquette

```
MATCH (n:airport) RETURN n
```

Notez que la première requête ci-dessus renvoie tous les nœuds du graphe, et les deux suivantes renvoient tous les nœuds ayant une relation, ce qui n'est généralement pas recommandé. Dans presque tous les cas, il est utile d'affiner les données renvoyées. Pour ce faire, spécifiez les étiquettes et les propriétés des nœuds ou des relations, comme dans le quatrième exemple.

Vous trouverez un aide-mémoire pratique pour la syntaxe d'openCypher dans le référentiel Github d'exemples Neptune.

Servlet de statut Neptune openCypher et point de terminaison de statut

Le point de terminaison de statut openCypher permet d'accéder aux informations relatives aux requêtes qui sont en cours d'exécution sur le serveur ou en attente d'exécution. Il vous permet également d'annuler ces requêtes. Le point de terminaison est :

```
https://(the server):(the port number)/openCypher/status
```

Vous pouvez utiliser les méthodes HTTP `GET` et `POST` pour obtenir le statut actuel à partir du serveur ou pour annuler une requête. Vous pouvez également utiliser la méthode `DELETE` pour annuler une requête en cours ou en attente.

Paramètres des demandes de statut

Paramètres des requêtes de statut

- **`includeWaiting`** (`trueoufalse`) : lorsque ce paramètre est défini sur `true` et que d'autres paramètres ne sont pas présents, des informations de statut pour les requêtes en attente ainsi que pour les requêtes en cours d'exécution sont renvoyées.
- **`cancelQuery`** : utilisé uniquement avec les méthodes `GET` et `POST` pour indiquer qu'il s'agit d'une demande d'annulation. La méthode `DELETE` n'a pas besoin de ce paramètre.

  La valeur du paramètre `cancelQuery` n'est pas utilisée, mais lorsque `cancelQuery` est présent, le paramètre `queryId` est obligatoire pour identifier la requête à annuler.
- **`queryId`** : contient l'ID d'une requête spécifique.

  Lorsque ce paramètre est utilisé avec la méthode `GET` ou `POST` et que le paramètre `cancelQuery` n'est pas présent, les informations de statut sont renvoyées par `queryId` pour la requête spécifique qu'il identifie. Si le paramètre `cancelQuery` est présent, la requête spécifique identifiée par `queryId` est annulée.

Lorsqu'il est utilisé avec la méthode `DELETE`, `queryId` indique toujours une requête spécifique à annuler.

- **`silent`** : utilisé uniquement lors de l'annulation d'une requête. Si ce paramètre est défini sur `true`, l'annulation se produit silencieusement.

Champs de réponse aux demandes de statut

Champs de réponse relatifs au statut si l'ID d'une requête spécifique n'est pas fourni

- accepté QueryCount — Le nombre de requêtes acceptées mais non encore terminées, y compris les requêtes dans la file d'attente.
- running QueryCount — Le nombre de requêtes OpenCypher en cours d'exécution.
- queries : requêtes openCypher actuelles.

Champs de réponse relatifs au statut pour une requête spécifique

- queryID : identifiant GUID de la requête. Neptune attribue automatiquement cette valeur d'ID à chaque requête, mais vous pouvez également attribuer votre propre ID (voir Injection d'un ID personnalisé dans une requête Neptune Gremlin ou SPARQL).
- queryString : requête soumise. Celle-ci est tronquée à 1 024 caractères si elle est plus longue que cela.
- query EvalStats — Statistiques pour cette requête :
  - waited : indique la durée d'attente de la requête, en millisecondes.
  - elapsed : nombre de microsecondes d'exécution de la requête jusqu'au moment T.
  - cancelled : `True` indique que la requête a été annulée, ou `False` qu'elle n'a pas été annulée.

Exemples de demandes de statut et de réponses

- Demande de statut de toutes les requêtes, y compris celles en attente :

  AWS CLI

```
aws neptunedata get-open-cypher-query-status \
  --endpoint-url https://your-neptune-endpoint:port \
  --include-waiting
```

Pour plus d'informations, consultez get-open-cypher-query-status dans le manuel de référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.get_open_cypher_query_status(
    includeWaiting=True
)

print(response)
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher/status \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "includeWaiting=true"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl https://your-neptune-endpoint:port/openCypher/status \
  --data-urlencode "includeWaiting=true"
```

Réponse :

```
{
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- Demande de statut des requêtes en cours, à l'exclusion de celles en attente :

  AWS CLI

  ```
  aws neptunedata get-open-cypher-query-status \
    --endpoint-url https://your-neptune-endpoint:port
  ```

  Pour plus d'informations, consultez get-open-cypher-query-status dans le manuel de référence des commandes. AWS CLI

  SDK

  ```
  import boto3
  from botocore.config import Config

  client = boto3.client(
      'neptunedata',
      endpoint_url='https://your-neptune-endpoint:port',
      config=Config(read_timeout=None, retries={'total_max_attempts': 1})
  )

  response = client.get_open_cypher_query_status()

  print(response)
  ```

  Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

  awscurl

  ```
  awscurl https://your-neptune-endpoint:port/openCypher/status \
    --region us-east-1 \
    --service neptune-db
  ```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl https://your-neptune-endpoint:port/openCypher/status
```

Réponse :

```
{
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- Demande de statut d'une seule requête :

  AWS CLI

  ```
  aws neptunedata get-open-cypher-query-status \
    --endpoint-url https://your-neptune-endpoint:port \
    --query-id eadc6eea-698b-4a2f-8554-5270ab17ebee
  ```

  Pour plus d'informations, consultez get-open-cypher-query-status dans le manuel de référence des commandes. AWS CLI

  SDK

  ```
  import boto3
  from botocore.config import Config

  client = boto3.client(
      'neptunedata',
      endpoint_url='https://your-neptune-endpoint:port',
      config=Config(read_timeout=None, retries={'total_max_attempts': 1})
  )

  response = client.get_open_cypher_query_status(
  ```

```
    queryId='eadc6eea-698b-4a2f-8554-5270ab17ebee'
)

print(response)
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher/status \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "queryId=eadc6eea-698b-4a2f-8554-5270ab17ebee"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl https://your-neptune-endpoint:port/openCypher/status \
  --data-urlencode "queryId=eadc6eea-698b-4a2f-8554-5270ab17ebee"
```

Réponse :

```
{
  "queryId" : "eadc6eea-698b-4a2f-8554-5270ab17ebee",
  "queryString" : "MATCH (n1)-[:knows]->(n2), (n2)-[:knows]->(n3), (n3)-[:knows]-
>(n4), (n4)-[:knows]->(n5), (n5)-[:knows]->(n6), (n6)-[:knows]->(n7), (n7)-[:knows]-
>(n8), (n8)-[:knows]->(n9), (n9)-[:knows]->(n10) RETURN COUNT(n1);",
  "queryEvalStats" : {
    "waited" : 0,
    "elapsed" : 23463,
    "cancelled" : false
  }
}
```

- Demandes d'annulation d'une requête

AWS CLI

```
aws neptunedata cancel-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --query-id f43ce17b-db01-4d37-a074-c76d1c26d7a9
```

Pour plus d'informations, voir cancel-open-cypher-query dans le manuel de référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.cancel_open_cypher_query(
    queryId='f43ce17b-db01-4d37-a074-c76d1c26d7a9'
)

print(response)
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher/status \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "cancelQuery" \
  -d "queryId=f43ce17b-db01-4d37-a074-c76d1c26d7a9"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

1. En utilisant `POST` :

```
curl -X POST https://your-neptune-endpoint:port/openCypher/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=f43ce17b-db01-4d37-a074-c76d1c26d7a9"
```

2. En utilisant `GET` :

```
curl -X GET https://your-neptune-endpoint:port/openCypher/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=588af350-cfde-4222-bee6-b9cedc87180d"
```

3. En utilisant `DELETE` :

```
curl -X DELETE \
  "https://your-neptune-endpoint:port/openCypher/status?queryId=b9a516d1-
d25c-4301-bb80-10b2743ecf0e"
```

Réponse :

```
{
  "status" : "200 OK",
  "payload" : true
}
```

Le point de terminaison HTTPS Amazon Neptune OpenCypher

Rubriques

- OpenCypher lire et écrire des requêtes sur le point de terminaison HTTPS
- Le format de résultats OpenCypher JSON par défaut
- En-têtes de suivi HTTP facultatifs pour les réponses en plusieurs parties OpenCypher

> **Note**
>
> Neptune ne prend actuellement pas en charge HTTP/2 les requêtes d'API REST. Les clients doivent l'utiliser HTTP/1.1 lorsqu'ils se connectent aux points de terminaison.

OpenCypher lire et écrire des requêtes sur le point de terminaison HTTPS

Le point de terminaison OpenCypher HTTPS prend en charge les requêtes de lecture et de mise à jour en utilisant à la fois la `POST` méthode `GET` et. Les méthodes `DELETE` et `PUT` ne sont pas prises en charge.

Les instructions suivantes vous indiquent comment vous connecter au point de OpenCypher terminaison à l'aide de la `curl` commande et du protocole HTTPS. Vous devez suivre ces instructions à partir d'une instance Amazon EC2 dans le même cloud privé virtuel (VPC) (VPC) que l'instance de base de données Neptune.

La syntaxe est la suivante :

```
HTTPS://(the server):(the port number)/openCypher
```

Voici un exemple de requête de lecture :

AWS CLI

```
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "MATCH (n1) RETURN n1"
```

Pour plus d'informations, voir execute-open-cypher-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
```

```
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_query(
    openCypherQuery='MATCH (n1) RETURN n1'
)

print(response['results'])
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "query=MATCH (n1) RETURN n1"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=MATCH (n1) RETURN n1"
```

Voici un exemple de write/update requête :

AWS CLI

```
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "CREATE (n:Person { age: 25 })"
```

Pour plus d'informations, voir execute-open-cypher-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_query(
    openCypherQuery='CREATE (n:Person { age: 25 })'
)

print(response['results'])
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "query=CREATE (n:Person { age: 25 })"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=CREATE (n:Person { age: 25 })"
```

Le format de résultats OpenCypher JSON par défaut

Le format JSON suivant est renvoyé par défaut ou en définissant explicitement l'en-tête de demande sur `Accept: application/json`. Ce format est conçu pour être facilement analysé en objets à l'aide des fonctionnalités du langage natif de la plupart des bibliothèques.

Le document JSON renvoyé contient un champ, `results`, qui comporte les valeurs renvoyées par la requête. Les exemples ci-dessous montrent la mise en forme JSON pour les valeurs courantes.

Exemple de réponse pour une valeur :

```
{
  "results": [
    {
      "count(a)": 121
    }
  ]
}
```

Exemple de réponse pour un nœud :

```
{
  "results": [
    {
      "a": {
        "~id": "22",
        "~entityType": "node",
        "~labels": [
          "airport"
        ],
        "~properties": {
          "desc": "Seattle-Tacoma",
          "lon": -122.30899810791,
          "runways": 3,
          "type": "airport",
          "country": "US",
          "region": "US-WA",
          "lat": 47.4490013122559,
          "elev": 432,
          "city": "Seattle",
          "icao": "KSEA",
          "code": "SEA",
          "longest": 11901
```

```
        }
      }
    }
  ]
}
```

Exemple de réponse pour une relation :

```
{
  "results": [
    {
      "r": {
        "~id": "7389",
        "~entityType": "relationship",
        "~start": "22",
        "~end": "151",
        "~type": "route",
        "~properties": {
          "dist": 956
        }
      }
    }
  ]
}
```

Exemple de réponse pour un chemin :

```
{
  "results": [
    {
      "p": [
        {
          "~id": "22",
          "~entityType": "node",
          "~labels": [
            "airport"
          ],
          "~properties": {
            "desc": "Seattle-Tacoma",
            "lon": -122.30899810791,
            "runways": 3,
            "type": "airport",
            "country": "US",
```

```
            "region": "US-WA",
            "lat": 47.4490013122559,
            "elev": 432,
            "city": "Seattle",
            "icao": "KSEA",
            "code": "SEA",
            "longest": 11901
          }
        },
        {
          "~id": "7389",
          "~entityType": "relationship",
          "~start": "22",
          "~end": "151",
          "~type": "route",
          "~properties": {
            "dist": 956
          }
        },
        {
          "~id": "151",
          "~entityType": "node",
          "~labels": [
            "airport"
          ],
          "~properties": {
            "desc": "Ontario International Airport",
            "lon": -117.600997924805,
            "runways": 2,
            "type": "airport",
            "country": "US",
            "region": "US-CA",
            "lat": 34.0559997558594,
            "elev": 944,
            "city": "Ontario",
            "icao": "KONT",
            "code": "ONT",
            "longest": 12198
          }
        }
      ]
    }
  ]
```

```
}
```

En-têtes de suivi HTTP facultatifs pour les réponses en plusieurs parties OpenCypher

Cette fonctionnalité est disponible à partir de la version 1.4.5.0 du moteur Neptune.

La réponse HTTP aux OpenCypher requêtes et aux mises à jour est généralement renvoyée en plusieurs parties. Lorsque des défaillances surviennent après l'envoi des segments de réponse initiaux (avec un code d'état HTTP de 200), il peut être difficile de diagnostiquer le problème. Par défaut, `Neptune signale ces défaillances en ajoutant un message d'erreur dans le corps du message, qui peut être corrompu en raison de la nature en streaming de la réponse.

Utilisation des en-têtes de suivi

Pour améliorer la détection et le diagnostic des erreurs, vous pouvez activer les en-têtes de suivi en incluant un en-tête de remorques à codage de transfert (TE) (te : trailers) dans votre demande. De cette manière, Neptune inclura deux nouveaux champs d'en-tête dans les en-têtes de suivi des fragments de réponse :

- `X-Neptune-Status`— contient le code de réponse suivi d'un nom court. Par exemple, en cas de réussite, l'en-tête final serait : `X-Neptune-Status: 200 OK`. En cas de panne, le code de réponse serait un code d'erreur du moteur Neptune tel que. `X-Neptune-Status: 500 TimeLimitExceededException`
- `X-Neptune-Detail`— est vide pour les demandes réussies. En cas d'erreur, il contient le message d'erreur JSON. Étant donné que seuls les caractères ASCII sont autorisés dans les valeurs d'en-tête HTTP, la chaîne JSON est encodée en URL. Le message d'erreur est également toujours ajouté au corps du message de réponse.

Pour plus d'informations, consultez la page MDN sur les en-têtes de requête TE.

OpenCypher exemple d'utilisation des en-têtes de suivi

Cet exemple montre comment les en-têtes de fin permettent de diagnostiquer une requête qui dépasse sa limite de temps :

```
curl --raw 'https://your-neptune-endpoint:port/openCypher' \
-H 'TE: trailers' \
-d 'query=MATCH(n) RETURN n.firstName'
```

```
Output:
< HTTP/1.1 200 OK
< transfer-encoding: chunked
< trailer: X-Neptune-Status, X-Neptune-Detail
< content-type: application/json;charset=UTF-8
<
<
{
  "results": [{
      "n.firstName": "Hossein"
    }, {
      "n.firstName": "Jan"
    }, {
      "n.firstName": "Miguel"
    }, {
      "n.firstName": "Eric"
    },
{"detailedMessage":"Operation terminated (deadline exceeded)",
"code":"TimeLimitExceededException",
"requestId":"a7e9d2aa-fbb7-486e-8447-2ef2a8544080",
"message":"Operation terminated (deadline exceeded)"}
0
X-Neptune-Status: 500 TimeLimitExceededException
X-Neptune-Detail: %7B%22detailedMessage%22%3A%22Operation+terminated+%28deadline
+exceeded%29%22%2C%22code%22%3A%22TimeLimitExceededException%22%2C%22requestId%22%3A
%22a7e9d2aa-fbb7-486e-8447-2ef2a8544080%22%2C%22message%22%3A%22Operation+terminated+
%28deadline+exceeded%29%22%7D
```

Répartition des réponses :

L'exemple précédent montre comment une OpenCypher réponse avec des en-têtes de fin peut aider à diagnostiquer les échecs de requête. Nous voyons ici quatre parties séquentielles : (1) les en-têtes initiaux avec un statut 200 OK indiquant le début du streaming, (2) les résultats JSON partiels (cassés) diffusés avec succès avant l'échec, (3) le message d'erreur ajouté indiquant le délai d'expiration, et (4) les en-têtes suivants contenant l'état final (500) et des informations d'erreur détaillées. TimeLimitExceededException

Utilisation de AWS SDK pour exécuter des requêtes OpenCypher

Avec le AWS SDK, vous pouvez exécuter des requêtes OpenCypher sur votre graphe Neptune en utilisant le langage de programmation de votre choix. Le SDK de l'API de données Neptune (nom

du service`neptunedata`) fournit l'ExecuteOpenCypherQueryaction permettant de soumettre des requêtes OpenCypher.

Vous devez exécuter ces exemples depuis une instance Amazon EC2 dans le même cloud privé virtuel (VPC) que votre cluster de base de données Neptune, ou depuis un emplacement disposant d'une connectivité réseau avec le point de terminaison de votre cluster.

Vous trouverez ci-dessous des liens directs vers la documentation de référence `neptunedata` de l'API pour le service dans chaque langue du SDK :

Langage de programmation	Référence de l'API neptunedata
C++	https://sdk.amazonaws.com/cpp/api/LATEST/aws-cpp-sdk-neptunedata/html/annotated.html
Go	https://docs.aws.amazon.com/sdk-for-go/api/service/neptunedata/
Java	https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/services/neptunedata/package-summary.html
JavaScript	https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/Package/-aws-sdk-client-neptunedata/
Kotlin	https://sdk.amazonaws.com/kotlin/api/latest/neptunedata/index.html
.NET	https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Neptunedata/NNeptunedata.html
PHP	https://docs.aws.amazon.com/aws-sdk-php/v3/api/namespace-Aws.Neptunedata.html
Python	https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/neptunedata.html

Langage de programmation	Référence de l'API neptunedata
Ruby	https://docs.aws.amazon.com/sdk-for-ruby/v3/api/Aws/Neptunedata.html
Rust	https://crates.io/crates/aws-sdk-neptunedata
INTERFACE DE LIGNE DE COMMANDE (CLI)	https://docs.aws.amazon.com/cli/latest/reference/neptunedata/

openCypher AWS Exemples de SDK

Les exemples suivants montrent comment configurer un `neptunedata` client, exécuter une requête OpenCypher et imprimer les résultats. Remplacez *`YOUR_NEPTUNE_HOST`* et *`YOUR_NEPTUNE_PORT`* par le point de terminaison et le port de votre cluster de base de données Neptune.

> ⚠ Client-side configuration du délai d'expiration et des nouvelles tentatives
>
> Le délai d'attente du client SDK contrôle le temps d'attente d'une réponse par le client. Il ne contrôle pas la durée d'exécution de la requête sur le serveur. Si le client expire avant la fin du serveur, la requête peut continuer à s'exécuter sur Neptune alors que le client n'a aucun moyen de récupérer les résultats.
> Nous vous recommandons de définir le délai de lecture côté client sur `0` (aucun délai d'attente) ou sur une valeur supérieure d'au moins quelques secondes au paramètre neptune_query_timeout côté serveur sur votre cluster de base de données Neptune. Cela permet à Neptune de contrôler l'expiration des requêtes.
> Nous recommandons également de fixer le nombre maximum de tentatives à 1 (aucune tentative). Si le SDK réessaie une requête toujours en cours d'exécution sur le serveur, cela peut entraîner des opérations dupliquées. Cela est particulièrement important pour les requêtes de mutation, où une nouvelle tentative peut entraîner des écritures dupliquées involontaires.

Python

1. Suivez les instructions d'installation pour installer Boto3.
2. Créez un fichier nommé `openCypherExample.py` et collez le code suivant :

```
import boto3
import json
from botocore.config import Config

# Disable the client-side read timeout and retries so that
# Neptune's server-side neptune_query_timeout controls query duration.
client = boto3.client(
    'neptunedata',
    endpoint_url=f'https://YOUR_NEPTUNE_HOST:YOUR_NEPTUNE_PORT',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_query(
    openCypherQuery='MATCH (n) RETURN n LIMIT 1'
)

print(json.dumps(response['results'], indent=2))
```

3. Exécutez l'exemple : `python openCypherExample.py`

Java

1. Suivez les instructions d'installation pour configurer le AWS SDK for Java.
2. Utilisez le code suivant pour configurer une`NeptunedataClient`, exécuter une requête OpenCypher et imprimer le résultat :

```
import java.net.URI;
import java.time.Duration;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.services.neptunedata.NeptunedataClient;
import
 software.amazon.awssdk.services.neptunedata.model.ExecuteOpenCypherQueryRequest;
import
 software.amazon.awssdk.services.neptunedata.model.ExecuteOpenCypherQueryResponse;

// Disable the client-side timeout and retries so that
// Neptune's server-side neptune_query_timeout controls query duration.
NeptunedataClient client = NeptunedataClient.builder()
    .endpointOverride(URI.create("https://YOUR_NEPTUNE_HOST:YOUR_NEPTUNE_PORT"))
    .overrideConfiguration(ClientOverrideConfiguration.builder()
```

```
        .apiCallTimeout(Duration.ZERO)
        .retryPolicy(RetryPolicy.none())
        .build())
    .build();

ExecuteOpenCypherQueryRequest request = ExecuteOpenCypherQueryRequest.builder()
    .openCypherQuery("MATCH (n) RETURN n LIMIT 1")
    .build();

ExecuteOpenCypherQueryResponse response = client.executeOpenCypherQuery(request);

System.out.println(response.results().toString());
```

JavaScript

1. Suivez les instructions d'installation pour configurer le AWS SDK pour JavaScript. Installez le package client neptunedata :. `npm install @aws-sdk/client-neptunedata`
2. Créez un fichier nommé `openCypherExample.js` et collez le code suivant :

```
import { NeptunedataClient, ExecuteOpenCypherQueryCommand } from "@aws-sdk/client-
neptunedata";
import { NodeHttpHandler } from "@smithy/node-http-handler";

const config = {
    endpoint: "https://YOUR_NEPTUNE_HOST:YOUR_NEPTUNE_PORT",
    // Disable the client-side request timeout so that
    // Neptune's server-side neptune_query_timeout controls query duration.
    requestHandler: new NodeHttpHandler({
        requestTimeout: 0
    }),
    maxAttempts: 1
};

const client = new NeptunedataClient(config);

const input = {
    openCypherQuery: "MATCH (n) RETURN n LIMIT 1"
};

const command = new ExecuteOpenCypherQueryCommand(input);
const response = await client.send(command);
```

```
console.log(JSON.stringify(response, null, 2));
```

3. Exécutez l'exemple : `node openCypherExample.js`

Utilisation du protocole Bolt pour envoyer des requêtes openCypher à Neptune

Bolt est un client/server protocole orienté instructions initialement développé par Neo4j et distribué sous licence Creative Commons 3.0. Attribution-ShareAlike Il est axé sur le client, ce qui signifie que le client initie toujours les échanges de messages.

Pour vous connecter à Neptune à l'aide des pilotes Bolt de Neo4j, remplacez simplement l'URL et le numéro de port par les points de terminaison de votre cluster à l'aide du schéma d'URI `bolt`. Si vous n'avez qu'une seule instance Neptune en cours d'exécution, utilisez le point de terminaison read_write. Si plusieurs instances sont en cours d'exécution, deux pilotes sont recommandés, l'un pour l'enregistreur et l'autre pour tous les réplicas en lecture. Si vous ne disposez que des deux points de terminaison par défaut, un pilote read_write et un pilote read_only sont suffisants, mais si vous avez également des points de terminaison personnalisés, envisagez de créer une instance de pilote pour chacun d'eux.

> **Note**
>
> Bien que la spécification Bolt indique que Bolt peut se connecter en utilisant TCP ou, WebSockets Neptune ne prend en charge que les connexions TCP pour Bolt.

Neptune autorise jusqu'à 1 000 connexions Bolt simultanées sur toutes les tailles d'instance, à l'exception de t3.medium et t4g.medium. Sur les instances t3.medium et t4g.medium, seules 512 connexions sont autorisées.

Pour obtenir des exemples de requêtes openCypher dans différents langages utilisant les pilotes Bolt, consultez les guides spécifiques aux pilotes et aux langages de Neo4j.

> **Important**
>
> Les pilotes Neo4j Bolt pour Python JavaScript, .NET et Golang ne prenaient pas initialement en charge le renouvellement automatique des jetons d'authentification AWS Signature v4. Autrement dit, après l'expiration de la signature (souvent au bout de cinq minutes), le pilote

ne parvenait pas à s'authentifier et les demandes suivantes échouaient. Les exemples Python JavaScript, .NET et Go ci-dessous étaient tous concernés par ce problème. Consultez le numéro #834 du pilote Python Neo4j, le numéro #664 du pilote Neo4j .NET, le problème #993 du pilote Neo4j et le numéro #429 JavaScript du pilote Neo4j GoLang pour plus d'informations.
À partir de la version 5.8.0 du pilote, une nouvelle version préliminaire de l'API de réauthentification a été publiée pour le pilote Go (voir v5.8.0 - Feedback wanted on re-authentication).

Utilisation de Bolt pour se connecter à Neptune

Vous pouvez télécharger un pilote pour la version que vous souhaitez utiliser à partir du référentiel Maven MVN, ou ajouter cette dépendance à votre projet :

```
<dependency>
  <groupId>org.neo4j.driver</groupId>
  <artifactId>neo4j-java-driver</artifactId>
  <version>4.3.3</version>
</dependency>
```

Ensuite, pour vous connecter à Neptune en Java à l'aide de l'un de ces pilotes Bolt, créez une instance de pilote pour l' primary/writer instance de votre cluster à l'aide d'un code tel que celui-ci :

```
import org.neo4j.driver.Driver;
import org.neo4j.driver.GraphDatabase;

final Driver driver =
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
    AuthTokens.none(),
    Config.builder().withEncryption()
                    .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                    .build());
```

Si vous possédez un ou plusieurs réplicas de lecteurs, vous pouvez également créer une instance de pilote pour ceux-ci à l'aide d'un code similaire à ce qui suit :

```
final Driver read_only_driver =              // (without connection timeout)
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
      Config.builder().withEncryption()
```

```
                    .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                    .build());
```

Ou, avec un délai d'expiration :

```
final Driver read_only_timeout_driver =      // (with connection timeout)
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
    Config.builder().withConnectionTimeout(30, TimeUnit.SECONDS)
                    .withEncryption()
                    .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                    .build());
```

Si vous avez des points de terminaison personnalisés, il peut également être intéressant de créer une instance de pilote pour chacun d'entre eux.

Exemple de requête openCypher en Python avec Bolt

Voici comment créer une requête openCypher en Python avec Bolt :

```
python -m pip install neo4j
```

```
from neo4j import GraphDatabase
uri = "bolt://(your cluster endpoint URL):(your cluster port)"
driver = GraphDatabase.driver(uri, auth=("username", "password"), encrypted=True)
```

Notez que les paramètres `auth` sont ignorés.

Exemple de requête openCypher dans NET avec Bolt

Pour effectuer une requête OpenCypher dans .NET à l'aide de Bolt, la première étape consiste à installer le pilote Neo4j à l'aide de. NuHet Pour passer des appels synchrones, utilisez la version `.Simple` comme suit :

```
Install-Package Neo4j.Driver.Simple-4.3.0
```

```
using Neo4j.Driver;

namespace hello
{
  // This example creates a node and reads a node in a Neptune
  // Cluster where IAM Authentication is not enabled.
```

```
 public class HelloWorldExample : IDisposable
 {
   private bool _disposed = false;
   private readonly IDriver _driver;
   private static string url = "bolt://(your cluster endpoint URL):(your cluster
port)";
   private static string createNodeQuery = "CREATE (a:Greeting) SET a.message =
'HelloWorldExample'";
   private static string readNodeQuery = "MATCH(n:Greeting) RETURN n.message";

   ~HelloWorldExample() => Dispose(false);

   public HelloWorldExample(string uri)
   {
     _driver = GraphDatabase.Driver(uri, AuthTokens.None, o =>
o.WithEncryptionLevel(EncryptionLevel.Encrypted));
   }

   public void createNode()
   {
     // Open a session
     using (var session = _driver.Session())
     {
        // Run the query in a write transaction
       var greeting = session.WriteTransaction(tx =>
       {
         var result = tx.Run(createNodeQuery);
         // Consume the result
         return result.Consume();
       });

       // The output will look like this:
       //   ResultSummary{Query=`CREATE (a:Greeting) SET a.message =
'HelloWorldExample".....
       Console.WriteLine(greeting);
     }
   }

   public void retrieveNode()
   {
     // Open a session
     using (var session = _driver.Session())
     {
       // Run the query in a read transaction
```

```
        var greeting = session.ReadTransaction(tx =>
        {
          var result = tx.Run(readNodeQuery);
          // Consume the result. Read the single node
          // created in a previous step.
          return result.Single()[0].As<string>();
        });
        // The output will look like this:
        //   HelloWorldExample
        Console.WriteLine(greeting);
      }
    }

    public void Dispose()
    {
      Dispose(true);
      GC.SuppressFinalize(this);
    }

    protected virtual void Dispose(bool disposing)
    {
      if (_disposed)
        return;
      if (disposing)
      {
        _driver?.Dispose();
      }
      _disposed = true;
    }

    public static void Main()
    {
      using (var apiCaller = new HelloWorldExample(url))
      {
        apiCaller.createNode();
        apiCaller.retrieveNode();
      }
    }
  }
}
```

Exemple de requête Java openCypher à l'aide de Bolt avec l'authentification IAM

Le code Java ci-dessous montre comment créer des requêtes openCypher en Java à l'aide de Bolt avec l'authentification IAM. Le JavaDoc commentaire décrit son utilisation. Une fois qu'une instance de pilote est disponible, vous pouvez l'utiliser pour effectuer plusieurs demandes authentifiées.

```
package software.amazon.neptune.bolt;

import com.amazonaws.DefaultRequest;
import com.amazonaws.Request;
import com.amazonaws.auth.AWS4Signer;
import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.http.HttpMethodName;
import com.google.gson.Gson;
import lombok.Builder;
import lombok.Getter;
import lombok.NonNull;
import org.neo4j.driver.Value;
import org.neo4j.driver.Values;
import org.neo4j.driver.internal.security.InternalAuthToken;
import org.neo4j.driver.internal.value.StringValue;

import java.net.URI;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;

import static com.amazonaws.auth.internal.SignerConstants.AUTHORIZATION;
import static com.amazonaws.auth.internal.SignerConstants.HOST;
import static com.amazonaws.auth.internal.SignerConstants.X_AMZ_DATE;
import static com.amazonaws.auth.internal.SignerConstants.X_AMZ_SECURITY_TOKEN;

/**
 * Use this class instead of `AuthTokens.basic` when working with an IAM
 * auth-enabled server. It works the same as `AuthTokens.basic` when using
 * static credentials, and avoids making requests with an expired signature
 * when using temporary credentials. Internally, it generates a new signature
 * on every invocation (this may change in a future implementation).
 *
 * Note that authentication happens only the first time for a pooled connection.
 *
 * Typical usage:
 *
```

```
 * NeptuneAuthToken authToken = NeptuneAuthToken.builder()
 *     .credentialsProvider(credentialsProvider)
 *     .region("aws region")
 *     .url("cluster endpoint url")
 *     .build();
 *
 * Driver driver = GraphDatabase.driver(
 *     authToken.getUrl(),
 *     authToken,
 *     config
 * );
 */

public class NeptuneAuthToken extends InternalAuthToken {
  private static final String SCHEME = "basic";
  private static final String REALM = "realm";
  private static final String SERVICE_NAME = "neptune-db";
  private static final String HTTP_METHOD_HDR = "HttpMethod";
  private static final String DUMMY_USERNAME = "username";
  @NonNull
  private final String region;
  @NonNull
  @Getter
  private final String url;
  @NonNull
  private final AWSCredentialsProvider credentialsProvider;
  private final Gson gson = new Gson();

  @Builder
  private NeptuneAuthToken(
      @NonNull final String region,
      @NonNull final String url,
      @NonNull final AWSCredentialsProvider credentialsProvider
  ) {
      // The superclass caches the result of toMap(), which we don't want
      super(Collections.emptyMap());
      this.region = region;
      this.url = url;
      this.credentialsProvider = credentialsProvider;
  }

  @Override
  public Map<String, Value> toMap() {
    final Map<String, Value> map = new HashMap<>();
```

```
    map.put(SCHEME_KEY, Values.value(SCHEME));
    map.put(PRINCIPAL_KEY, Values.value(DUMMY_USERNAME));
    map.put(CREDENTIALS_KEY, new StringValue(getSignedHeader()));
    map.put(REALM_KEY, Values.value(REALM));

    return map;
  }

  private String getSignedHeader() {
    final Request<Void> request = new DefaultRequest<>(SERVICE_NAME);
    request.setHttpMethod(HttpMethodName.GET);
    request.setEndpoint(URI.create(url));
    // Comment out the following line if you're using an engine version older than
 1.2.0.0
    request.setResourcePath("/opencypher");

    final AWS4Signer signer = new AWS4Signer();
    signer.setRegionName(region);
    signer.setServiceName(request.getServiceName());
    signer.sign(request, credentialsProvider.getCredentials());

    return getAuthInfoJson(request);
  }

  private String getAuthInfoJson(final Request<Void> request) {
    final Map<String, Object> obj = new HashMap<>();
    obj.put(AUTHORIZATION, request.getHeaders().get(AUTHORIZATION));
    obj.put(HTTP_METHOD_HDR, request.getHttpMethod());
    obj.put(X_AMZ_DATE, request.getHeaders().get(X_AMZ_DATE));
    obj.put(HOST, request.getHeaders().get(HOST));
    obj.put(X_AMZ_SECURITY_TOKEN, request.getHeaders().get(X_AMZ_SECURITY_TOKEN));

    return gson.toJson(obj);
  }
}
```

Exemple de requête Python openCypher à l'aide de Bolt avec l'authentification IAM

La classe Python ci-dessous vous permet d'effectuer des requêtes openCypher en Python à l'aide de Bolt avec l'authentification IAM :

```
import json
```

```
from neo4j import Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import Credentials
from botocore.auth import (
  SigV4Auth,
  _host_from_url,
)

SCHEME = "basic"
REALM = "realm"
SERVICE_NAME = "neptune-db"
DUMMY_USERNAME = "username"
HTTP_METHOD_HDR = "HttpMethod"
HTTP_METHOD = "GET"
AUTHORIZATION = "Authorization"
X_AMZ_DATE = "X-Amz-Date"
X_AMZ_SECURITY_TOKEN = "X-Amz-Security-Token"
HOST = "Host"


class NeptuneAuthToken(Auth):
  def __init__(
    self,
    credentials: Credentials,
    region: str,
    url: str,
    **parameters
  ):
    # Do NOT add "/opencypher" in the line below if you're using an engine version
 older than 1.2.0.0
    request = AWSRequest(method=HTTP_METHOD, url=url + "/opencypher")
    request.headers.add_header("Host", _host_from_url(request.url))
    sigv4 = SigV4Auth(credentials, SERVICE_NAME, region)
    sigv4.add_auth(request)

    auth_obj = {
      hdr: request.headers[hdr]
      for hdr in [AUTHORIZATION, X_AMZ_DATE, X_AMZ_SECURITY_TOKEN, HOST]
    }
    auth_obj[HTTP_METHOD_HDR] = request.method
    creds: str = json.dumps(auth_obj)
    super().__init__(SCHEME, DUMMY_USERNAME, creds, REALM, **parameters)
```

Utilisez cette classe pour créer un pilote comme suit :

```
  authToken = NeptuneAuthToken(creds, REGION, URL)
  driver = GraphDatabase.driver(URL, auth=authToken, encrypted=True)
```

Node.js Exemple d'utilisation de l'authentification IAM et de Bolt

Le Node.js code ci-dessous utilise le AWS SDK pour la JavaScript version 3 et la syntaxe ES6 pour créer un pilote qui authentifie les demandes :

```
import neo4j from "neo4j-driver";
import { HttpRequest }  from "@smithy/protocol-http";
import { defaultProvider } from "@aws-sdk/credential-provider-node";
import { SignatureV4 } from "@smithy/signature-v4";
import crypto from "@aws-crypto/sha256-js";
const { Sha256 } = crypto;
import assert from "node:assert";

const region = "us-west-2";
const serviceName = "neptune-db";
const host = "(your cluster endpoint URL)";
const port = 8182;
const protocol = "bolt";
const hostPort = host + ":" + port;
const url = protocol + "://" + hostPort;
const createQuery = "CREATE (n:Greeting {message: 'Hello'}) RETURN ID(n)";
const readQuery = "MATCH(n:Greeting) WHERE ID(n) = $id RETURN n.message";

async function signedHeader() {
  const req = new HttpRequest({
    method: "GET",
    protocol: protocol,
    hostname: host,
    port: port,
    // Comment out the following line if you're using an engine version older than
 1.2.0.0
    path: "/opencypher",
    headers: {
      host: hostPort
    }
  });

  const signer = new SignatureV4({
```

```
    credentials: defaultProvider(),
    region: region,
    service: serviceName,
    sha256: Sha256
  });

  return signer.sign(req, { unsignableHeaders: new Set(["x-amz-content-sha256"]) })
    .then((signedRequest) => {
      const authInfo = {
        "Authorization": signedRequest.headers["authorization"],
        "HttpMethod": signedRequest.method,
        "X-Amz-Date": signedRequest.headers["x-amz-date"],
        "Host": signedRequest.headers["host"],
        "X-Amz-Security-Token": signedRequest.headers["x-amz-security-token"]
      };
      return JSON.stringify(authInfo);
    });
}

async function createDriver() {
  let authToken = { scheme: "basic", realm: "realm", principal: "username",
 credentials: await signedHeader() };

  return neo4j.driver(url, authToken, {
      encrypted: "ENCRYPTION_ON",
      trust: "TRUST_SYSTEM_CA_SIGNED_CERTIFICATES",
      maxConnectionPoolSize: 1,
      // logging: neo4j.logging.console("debug")
    }
  );
}

async function unmanagedTxn(driver) {
  const session = driver.session();
  const tx = session.beginTransaction();
  try {
    const created = await tx.run(createQuery);
    const matched = await tx.run(readQuery, { id: created.records[0].get(0) });
    const msg = matched.records[0].get("n.message");
    assert.equal(msg, "Hello");
    await tx.commit();
  } catch (err) {
    // The transaction will be rolled back, now handle the error.
    console.log(err);
```

```
  } finally {
    await session.close();
  }
}

const driver = await createDriver();
try {
  await unmanagedTxn(driver);
} catch (err) {
  console.log(err);
} finally {
  await driver.close();
}
```

Exemple de requête .NET openCypher à l'aide de Bolt avec l'authentification IAM

Pour activer l'authentification IAM en .NET, vous devez signer une demande lors de l'établissement de la connexion. L'exemple ci-dessous montre comment créer un assistant `NeptuneAuthToken` pour générer un jeton d'authentification :

```
using Amazon.Runtime;
using Amazon.Util;
using Neo4j.Driver;
using System.Security.Cryptography;
using System.Text;
using System.Text.Json;
using System.Web;

namespace Hello
{
  /*
   * Use this class instead of `AuthTokens.None` when working with an IAM-auth-enabled
 server.
   *
   * Note that authentication happens only the first time for a pooled connection.
   *
   * Typical usage:
   *
   * var authToken = new NeptuneAuthToken(AccessKey, SecretKey,
 Region).GetAuthToken(Host);
   * _driver = GraphDatabase.Driver(Url, authToken, o =>
 o.WithEncryptionLevel(EncryptionLevel.Encrypted));
   */
```

```
 public class NeptuneAuthToken
 {
   private const string ServiceName = "neptune-db";
   private const string Scheme = "basic";
   private const string Realm = "realm";
   private const string DummyUserName = "username";
   private const string Algorithm = "AWS4-HMAC-SHA256";
   private const string AWSRequest = "aws4_request";

   private readonly string _accessKey;
   private readonly string _secretKey;
   private readonly string _region;

   private readonly string _emptyPayloadHash;

   private readonly SHA256 _sha256;


   public NeptuneAuthToken(string awsKey = null, string secretKey = null, string
region = null)
   {
     var awsCredentials = awsKey == null || secretKey == null
       ? FallbackCredentialsFactory.GetCredentials().GetCredentials()
       : null;

     _accessKey = awsKey ?? awsCredentials.AccessKey;
     _secretKey = secretKey ?? awsCredentials.SecretKey;
     _region = region ?? FallbackRegionFactory.GetRegionEndpoint().SystemName; //ex:
us-east-1

     _sha256 = SHA256.Create();
     _emptyPayloadHash = Hash(Array.Empty<byte>());
   }

   public IAuthToken GetAuthToken(string url)
   {
     return AuthTokens.Custom(DummyUserName, GetCredentials(url), Realm, Scheme);
   }

   /******************** AWS SIGNING FUNCTIONS ********************/
   private string Hash(byte[] bytesToHash)
   {
     return ToHexString(_sha256.ComputeHash(bytesToHash));
```

```
    }

    private static byte[] HmacSHA256(byte[] key, string data)
    {
      return new HMACSHA256(key).ComputeHash(Encoding.UTF8.GetBytes(data));
    }

    private byte[] GetSignatureKey(string dateStamp)
    {
      var kSecret = Encoding.UTF8.GetBytes($"AWS4{_secretKey}");
      var kDate = HmacSHA256(kSecret, dateStamp);
      var kRegion = HmacSHA256(kDate, _region);
      var kService = HmacSHA256(kRegion, ServiceName);
      return HmacSHA256(kService, AWSRequest);
    }

    private static string ToHexString(byte[] array)
    {
      return Convert.ToHexString(array).ToLowerInvariant();
    }

    private string GetCredentials(string url)
    {
      var request = new HttpRequestMessage
      {
        Method = HttpMethod.Get,
        RequestUri = new Uri($"https://{url}/opencypher")
      };

      var signedrequest = Sign(request);

      var headers = new Dictionary<string, object>
      {
        [HeaderKeys.AuthorizationHeader] =
signedrequest.Headers.GetValues(HeaderKeys.AuthorizationHeader).FirstOrDefault(),
        ["HttpMethod"] = HttpMethod.Get.ToString(),
        [HeaderKeys.XAmzDateHeader] =
signedrequest.Headers.GetValues(HeaderKeys.XAmzDateHeader).FirstOrDefault(),
        // Host should be capitalized, not like in Amazon.Util.HeaderKeys.HostHeader
        ["Host"] =
signedrequest.Headers.GetValues(HeaderKeys.HostHeader).FirstOrDefault(),
      };

      return JsonSerializer.Serialize(headers);
```

```
    }

    private HttpRequestMessage Sign(HttpRequestMessage request)
    {
      var now = DateTimeOffset.UtcNow;
      var amzdate = now.ToString("yyyyMMddTHHmmssZ");
      var datestamp = now.ToString("yyyyMMdd");

      if (request.Headers.Host == null)
      {
        request.Headers.Host = $"{request.RequestUri.Host}:{request.RequestUri.Port}";
      }

      request.Headers.Add(HeaderKeys.XAmzDateHeader, amzdate);

      var canonicalQueryParams = GetCanonicalQueryParams(request);

      var canonicalRequest = new StringBuilder();
      canonicalRequest.Append(request.Method + "\n");
      canonicalRequest.Append(request.RequestUri.AbsolutePath + "\n");
      canonicalRequest.Append(canonicalQueryParams + "\n");

      var signedHeadersList = new List<string>();
      foreach (var header in request.Headers.OrderBy(a => a.Key.ToLowerInvariant()))
      {
        canonicalRequest.Append(header.Key.ToLowerInvariant());
        canonicalRequest.Append(':');
        canonicalRequest.Append(string.Join(",", header.Value.Select(s => s.Trim())));
        canonicalRequest.Append('\n');
        signedHeadersList.Add(header.Key.ToLowerInvariant());
      }
      canonicalRequest.Append('\n');

      var signedHeaders = string.Join(";", signedHeadersList);
      canonicalRequest.Append(signedHeaders + "\n");
      canonicalRequest.Append(_emptyPayloadHash);

      var credentialScope = $"{datestamp}/{_region}/{ServiceName}/{AWSRequest}";
      var stringToSign = $"{Algorithm}\n{amzdate}\n{credentialScope}\n"
        + Hash(Encoding.UTF8.GetBytes(canonicalRequest.ToString()));

      var signing_key = GetSignatureKey(datestamp);
      var signature = ToHexString(HmacSHA256(signing_key, stringToSign));
```

```
      request.Headers.TryAddWithoutValidation(HeaderKeys.AuthorizationHeader,
        $"{Algorithm} Credential={_accessKey}/{credentialScope},
 SignedHeaders={signedHeaders}, Signature={signature}");

      return request;
    }

    private static string GetCanonicalQueryParams(HttpRequestMessage request)
    {
      var querystring = HttpUtility.ParseQueryString(request.RequestUri.Query);

      // Query params must be escaped in upper case (i.e. "%2C", not "%2c").
      var queryParams = querystring.AllKeys.OrderBy(a => a)
        .Select(key => $"{key}={Uri.EscapeDataString(querystring[key])}");
      return string.Join("&", queryParams);
    }
  }
}
```

Voici comment créer une requête openCypher en .NET à l'aide de Bolt avec l'authentification IAM. L'exemple ci-dessous utilise l'assistant `NeptuneAuthToken` :

```
using Neo4j.Driver;

namespace Hello
{
  public class HelloWorldExample
  {
    private const string Host = "(your hostname):8182";
    private const string Url = $"bolt://{Host}";
    private const string CreateNodeQuery = "CREATE (a:Greeting) SET a.message =
 'HelloWorldExample'";
    private const string ReadNodeQuery = "MATCH(n:Greeting) RETURN n.message";

    private const string AccessKey = "(your access key)";
    private const string SecretKey = "(your secret key)";
    private const string Region = "(your AWS region)"; // e.g. "us-west-2"

    private readonly IDriver _driver;

    public HelloWorldExample()
    {
```

```
      var authToken = new NeptuneAuthToken(AccessKey, SecretKey,
Region).GetAuthToken(Host);

      // Note that when the connection is reinitialized after max connection lifetime
      // has been reached, the signature token could have already been expired (usually
5 min)
      // You can face exceptions like:
      //   `Unexpected server exception 'Signature expired: XXXX is now earlier than
YYYY (ZZZZ - 5 min.)`
      _driver = GraphDatabase.Driver(Url, authToken, o =>

o.WithMaxConnectionLifetime(TimeSpan.FromMinutes(60)).WithEncryptionLevel(EncryptionLevel.Encr
   }

   public async Task CreateNode()
   {
     // Open a session
     using (var session = _driver.AsyncSession())
     {
       // Run the query in a write transaction
       var greeting = await session.WriteTransactionAsync(async tx =>
       {
         var result = await tx.RunAsync(CreateNodeQuery);
         // Consume the result
         return await result.ConsumeAsync();
       });

       // The output will look like this:
       //   ResultSummary{Query=`CREATE (a:Greeting) SET a.message =
'HelloWorldExample".....
       Console.WriteLine(greeting.Query);
     }
   }

   public async Task RetrieveNode()
   {
     // Open a session
     using (var session = _driver.AsyncSession())
     {
       // Run the query in a read transaction
       var greeting = await session.ReadTransactionAsync(async tx =>
       {
         var result = await tx.RunAsync(ReadNodeQuery);
         var records = await result.ToListAsync();
```

```
          // Consume the result. Read the single node
          // created in a previous step.
          return records[0].Values.First().Value;
        });
        // The output will look like this:
        //   HelloWorldExample
        Console.WriteLine(greeting);
      }
    }
  }
}
```

Cet exemple peut être lancé en exécutant le code ci-dessous sur `.NET 6` ou `.NET 7` avec les packages suivants :

- **`Neo4j`**`.Driver=4.3.0`
- **`AWSSDK`**`.Core=3.7.102.1`

```
namespace Hello
{
  class Program
  {
    static async Task Main()
    {
      var apiCaller = new HelloWorldExample();

      await apiCaller.CreateNode();
      await apiCaller.RetrieveNode();
    }
  }
}
```

Exemple de requête Golang openCypher à l'aide de Bolt avec l'authentification IAM

L'exemple suivant montre comment effectuer des requêtes OpenCypher dans Go à l'aide du protocole Bolt avec authentification IAM. Il utilise le SDK AWS pour Go v2 pour la signature Sigv4 et le pilote Neo4j Go v5 avec `AuthTokenManager` une structure qui implémente l'interface de gestion de jetons du pilote Neo4j Go `github.com/neo4j/neo4j-go-driver/v5/neo4j/`

`auth.TokenManager` () pour actualiser automatiquement les informations d'identification avant leur expiration.

Tout d'abord, créez une `AuthTokenManager` annonce qui génère SigV4-signed des jetons. Enregistrez ceci sous le nom `auth_token_manager.go` :

```
// AuthTokenManager for Amazon Neptune IAM authentication via the Bolt protocol.
// Provides SigV4-signed credentials to the Neo4j driver's auth interface.
package main

import (
 "context"
 "encoding/json"
 "fmt"
 "net/http"
 "reflect"
 "sync"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/credentials/stscreds"
 "github.com/aws/aws-sdk-go-v2/service/sts"
 "github.com/neo4j/neo4j-go-driver/v5/neo4j"
 "github.com/neo4j/neo4j-go-driver/v5/neo4j/db"
)

const (
 serviceName      = "neptune-db"
 emptyPayloadHash = "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855"
)

// AuthTokenManager manages SigV4-signed authentication tokens for Neptune with
 automatic refresh.
type AuthTokenManager struct {
 region          string
 endpoint        string
 refreshInterval time.Duration
 credentials     aws.CredentialsProvider
 mutex           sync.Mutex
 cachedToken     neo4j.AuthToken
 tokenTime       time.Time
```

```
}

// NewAuthTokenManager creates a new AuthTokenManager.
//
// Parameters:
//   - region: AWS region (e.g., "us-east-1")
//   - endpoint: Neptune endpoint with port (e.g.,
 "cluster.region.neptune.amazonaws.com:8182")
//   - profile: AWS profile name (optional, pass "" to use default)
//   - roleArn: AWS role ARN to assume (optional, pass "" to skip)
//   - refreshInterval: Token refresh interval
func NewAuthTokenManager(region, endpoint, profile, roleArn string, refreshInterval
 time.Duration) (*AuthTokenManager, error) {
 credentials, err := createCredentialsProvider(region, profile, roleArn)
 if err != nil {
  return nil, err
 }
 return &AuthTokenManager{
  region:          region,
  endpoint:        endpoint,
  refreshInterval: refreshInterval,
  credentials:     credentials,
 }, nil
}

// createCredentialsProvider builds an AWS credentials provider, optionally using
// a named profile and/or assuming a role.
func createCredentialsProvider(region, profile, roleArn string)
 (aws.CredentialsProvider, error) {
 ctx := context.Background()
 var opts []func(*config.LoadOptions) error
 opts = append(opts, config.WithRegion(region))
 if profile != "" {
  opts = append(opts, config.WithSharedConfigProfile(profile))
 }
 cfg, err := config.LoadDefaultConfig(ctx, opts...)
 if err != nil {
  return nil, fmt.Errorf("failed to load AWS config: %w", err)
 }
 var credentials aws.CredentialsProvider = cfg.Credentials
 if roleArn != "" {
  stsClient := sts.NewFromConfig(cfg)
  credentials = stscreds.NewAssumeRoleProvider(stsClient, roleArn, func(o
 *stscreds.AssumeRoleOptions) {
```

```
    o.RoleSessionName = "NeptuneAuthSession"
    o.Duration = 900 * time.Second
   })
  }
  return credentials, nil
}

// GetAuthToken returns a valid authentication token, using cached token if still
  valid.
func (m *AuthTokenManager) GetAuthToken(ctx context.Context) (neo4j.AuthToken, error) {
  m.mutex.Lock()
  defer m.mutex.Unlock()

  if time.Since(m.tokenTime) < m.refreshInterval && m.cachedToken.Tokens != nil {
   return m.cachedToken, nil
  }

  token, err := m.generateToken(ctx)
  if err != nil {
   return neo4j.AuthToken{}, err
  }

  m.cachedToken = token
  m.tokenTime = time.Now()
  return token, nil
}

// HandleSecurityException handles security exceptions by invalidating the cached
// token (if it matches the token that caused the error) and returning true so
// the driver retries with a fresh token. The comparison prevents a concurrent
// retry from unnecessarily invalidating a freshly generated token.
func (m *AuthTokenManager) HandleSecurityException(ctx context.Context, token
  neo4j.AuthToken, err *db.Neo4jError) (bool, error) {
  m.mutex.Lock()
  defer m.mutex.Unlock()
  if reflect.DeepEqual(m.cachedToken.Tokens, token.Tokens) {
   m.tokenTime = time.Time{}
   m.cachedToken = neo4j.AuthToken{}
  }
  return true, nil
}

// generateToken generates a new SigV4-signed authentication token for Neptune.
```

```
func (m *AuthTokenManager) generateToken(ctx context.Context) (neo4j.AuthToken, error)
 {
 req, err := http.NewRequest(http.MethodGet, "https://"+m.endpoint+"/opencypher", nil)
 if err != nil {
  return neo4j.AuthToken{}, err
 }
 req.Host = m.endpoint

 creds, err := m.credentials.Retrieve(ctx)
 if err != nil {
  return neo4j.AuthToken{}, err
 }

 signer := v4.NewSigner()
 err = signer.SignHTTP(ctx, creds, req, emptyPayloadHash, serviceName, m.region,
 time.Now())
 if err != nil {
  return neo4j.AuthToken{}, err
 }

 authData := map[string]string{
  "Authorization": req.Header.Get("Authorization"),
  "X-Amz-Date":    req.Header.Get("X-Amz-Date"),
  "Host":          m.endpoint,
  "HttpMethod":    req.Method,
 }

 if st := req.Header.Get("X-Amz-Security-Token"); st != "" {
  authData["X-Amz-Security-Token"] = st
 }

 authJSON, err := json.Marshal(authData)
 if err != nil {
  return neo4j.AuthToken{}, err
 }

 return neo4j.BasicAuth("username", string(authJSON), ""), nil
}
```

Utilisez ensuite le gestionnaire de jetons pour créer un pilote et rechercher un nœud par identifiant. Notez l'utilisation d'une requête paramétrée (`$nodeId`) au lieu d'une interpolation de chaîne :

```
package main
```

```
import (
 "context"
 "fmt"
 "log"
 "time"

 "github.com/neo4j/neo4j-go-driver/v5/neo4j"
)

func findNode(ctx context.Context, driver neo4j.DriverWithContext, nodeId string)
 (string, error) {
 session := driver.NewSession(ctx, neo4j.SessionConfig{
  AccessMode: neo4j.AccessModeRead,
 })
 defer session.Close(ctx)

 // Use parameterized queries to prevent injection and enable query plan caching.
 result, err := session.Run(ctx,
  "MATCH (n) WHERE ID(n) = $nodeId RETURN n",
  map[string]any{"nodeId": nodeId},
 )
 if err != nil {
  return "", fmt.Errorf("error running query: %v", err)
 }

 if !result.Next(ctx) {
  if err = result.Err(); err != nil {
   return "", fmt.Errorf("error fetching result: %v", err)
  }
  return "", fmt.Errorf("node not found")
 }

 n, found := result.Record().Get("n")
 if !found {
  return "", fmt.Errorf("node not found")
 }

 return fmt.Sprintf("%+v", n), nil
}

func main() {
 region := "(your AWS region)"                  // e.g. "us-east-1"
```

```
 endpoint := "(your Neptune endpoint):8182"     // e.g. "cluster.xxx.us-
east-1.neptune.amazonaws.com:8182"

 ctx := context.Background()

 // Pass a profile name for local development, or a role ARN for cross-account access
 authManager, err := NewAuthTokenManager(region, endpoint, "", "", 4*time.Minute) //
 Refresh before the 5-minute SigV4 signature expiry
 if err != nil {
  log.Fatalf("auth manager error: %v", err)
 }

 // bolt+s:// enables TLS with full certificate verification.
 // If you're developing on macOS, use bolt+ssc:// instead. Go on macOS uses the
 // system TLS verifier, which requires Certificate Transparency compliance that
 // Neptune endpoints don't support. In production (typically Linux), bolt+s://
 // works with system CA certificates.
 driver, err := neo4j.NewDriverWithContext("bolt+s://"+endpoint, authManager)
 if err != nil {
  log.Fatalf("driver error: %v", err)
 }
 defer driver.Close(ctx)

 if err = driver.VerifyConnectivity(ctx); err != nil {
  log.Fatalf("connectivity error: %v", err)
 }

 // Neptune assigns UUID-format node IDs if a user does not supply their own node IDs
 res, err := findNode(ctx, driver, "72c2e8c1-7d5f-5f30-10ca-9d2bb8c4afbc")
 if err != nil {
  log.Fatal(err)
 }
 fmt.Println(res)
}
```

> **Note**
>
> L'**`auth.TokenManager`**interface (**`github.com/neo4j/neo4j-go-driver/v5/neo4j/auth`**) utilisée dans cet exemple est devenue généralement disponible dans le pilote Neo4j Go v5.14.0. Cette interface permet l'actualisation automatique des informations d'identification, ce qui est nécessaire pour l'authentification Neptune IAM car les signatures

SigV4 ne sont valides que pendant une courte période et doivent être régénérées lorsque le pilote établit de nouvelles connexions.

Cet exemple a été validé à l'aide des modules Go suivants :

```
require (
    github.com/aws/aws-sdk-go-v2         v1.30.3
    github.com/aws/aws-sdk-go-v2/config   v1.27.27
    github.com/aws/aws-sdk-go-v2/credentials v1.17.27
    github.com/aws/aws-sdk-go-v2/service/sts v1.30.3
    github.com/neo4j/neo4j-go-driver/v5   v5.22.0
  )
```

Comportement des connexions Bolt dans Neptune

Voici quelques éléments à garder à l'esprit concernant les connexions Neptune Bolt :

- Comme les connexions Bolt sont créées au niveau de la couche TCP, vous ne pouvez pas utiliser un Application Load Balancer devant elles, comme c'est le cas avec un point de terminaison HTTP.
- Le port que Neptune utilise pour les connexions Bolt est le port de votre cluster de bases de données.
- Sur la base du préambule Bolt qui lui a été transmis, le serveur Neptune sélectionne la version Bolt la plus appropriée (1, 2, 3 ou 4.0).
- Le nombre maximum de connexions au serveur Neptune qu'un client peut ouvrir à tout moment est de 1 000.
- Si le client ne ferme pas la connexion après une requête, celle-ci peut être utilisée pour exécuter la requête suivante.
- Toutefois, si une connexion est inactive pendant 20 minutes, le serveur la ferme automatiquement.
- Si l'authentification IAM n'est pas activée, vous pouvez utiliser `AuthTokens.none()` au lieu de les fournir un nom d'utilisateur et un mot de passe factices. Par exemple, en Java :

```
GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
 AuthTokens.none(),

 Config.builder().withEncryption().withTrustStrategy(TrustStrategy.trustSystemCertificates())
```

- Lorsque l'authentification IAM est activée, une connexion Bolt est toujours déconnectée quelques minutes de plus que 10 jours après son établissement si elle n'a pas déjà été fermée pour une autre raison.
- Si le client envoie une requête à exécuter via une connexion sans avoir consommé les résultats d'une requête précédente, la nouvelle requête est supprimée. Pour ignorer les résultats précédents, le client doit envoyer un message de réinitialisation via la connexion.
- Une seule transaction à la fois peut être créée sur une connexion donnée.
- Si une exception se produit au cours d'une transaction, le serveur Neptune annule cette transaction et ferme la connexion. Dans ce cas, le pilote crée une autre connexion pour la prochaine requête.
- Sachez que les sessions ne sont pas adaptées aux threads. Diverses opérations parallèles doivent utiliser diverses sessions distinctes.

Exemples de requêtes paramétrées openCypher

Neptune prend en charge les requêtes openCypher paramétrées. Vous pouvez ainsi utiliser la même structure de requête plusieurs fois avec des arguments différents. Comme la structure de la requête ne change pas, Neptune peut mettre en cache son arbre syntaxique abstrait (AST) au lieu d'avoir à l'analyser plusieurs fois.

Exemple de requête paramétrée openCypher avec le point de terminaison HTTPS

Vous trouverez ci-dessous un exemple d'utilisation d'une requête paramétrée avec le point de terminaison HTTPS Neptune openCypher. Voici cette requête :

```
MATCH (n {name: $name, age: $age})
RETURN n
```

Les paramètres sont définis, comme suit :

```
parameters={"name": "john", "age": 20}
```

Vous pouvez soumettre la requête paramétrée comme suit :

AWS CLI

```
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "MATCH (n {name: \$name, age: \$age}) RETURN n" \
```

```
  --parameters '{"name": "john", "age": 20}'
```

Pour plus d'informations, voir execute-open-cypher-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_query(
    openCypherQuery='MATCH (n {name: $name, age: $age}) RETURN n',
    parameters='{"name": "john", "age": 20}'
)

print(response['results'])
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "query=MATCH (n {name: \$name, age: \$age}) RETURN n" \
  -d 'parameters={"name": "john", "age": 20}'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

En utilisant POST :

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=MATCH (n {name: \$name, age: \$age}) RETURN n" \
  -d "parameters={\"name\": \"john\", \"age\": 20}"
```

En utilisant `GET` (URL-encoded) :

```
curl -X GET \
  "https://your-neptune-endpoint:port/openCypher?query=MATCH%20%28n%20%7Bname:
\$name,age:\$age%7D%29%20RETURN%20n&parameters=%7B%22name%22:%22john%22,%22age
%22:20%7D"
```

En utilisant `DIRECT POST` :

```
curl -H "Content-Type: application/opencypher" \
  "https://your-neptune-endpoint:port/openCypher?parameters=%7B%22name%22:%22john
%22,%22age%22:20%7D" \
  -d "MATCH (n {name: \$name, age: \$age}) RETURN n"
```

Exemples de requêtes paramétrées openCypher avec Bolt

Voici un exemple Python d'une requête paramétrée openCypher avec le protocole Bolt :

```
from neo4j import GraphDatabase
uri = "bolt://[neptune-endpoint-url]:8182"
driver = GraphDatabase.driver(uri, auth=("", ""))

def match_name_and_age(tx, name, age):
  # Parameterized Query
  tx.run("MATCH (n {name: $name, age: $age}) RETURN n", name=name, age=age)

with driver.session() as session:
  # Parameters
  session.read_transaction(match_name_and_age, "john", 20)

driver.close()
```

Voici un exemple Java d'une requête paramétrée openCypher avec le protocole Bolt :

```
Driver driver = GraphDatabase.driver("bolt+s://(your cluster endpoint URL):8182");
HashMap<String, Object> parameters = new HashMap<>();
```

```
parameters.put("name", "john");
parameters.put("age", 20);
String queryString = "MATCH (n {name: $name, age: $age}) RETURN n";
Result result = driver.session().run(queryString, parameters);
```

Modèle de données openCypher

Le moteur Neptune openCypher s'appuie sur le même modèle de graphe de propriétés que Gremlin. En particulier :

- Chaque nœud a une ou plusieurs étiquettes. Si vous insérez un nœud sans étiquette, une étiquette par défaut nommée `vertex` est attachée. Si vous essayez de supprimer toutes les étiquettes d'un nœud, une erreur est générée.
- Une relation est une entité qui possède exactement un seul type de relation et qui forme une connexion unidirectionnelle entre deux nœuds (c'est-à-dire d'un nœud source à un nœud cible).
- Les nœuds et les relations peuvent avoir des propriétés, mais ce n'est pas obligatoire. Neptune prend en charge les nœuds et les relations n'ayant aucune propriété.
- Neptune ne prend pas en charge les métapropriétés, qui ne sont pas non plus incluses dans la spécification openCypher.
- Les propriétés du graphe peuvent avoir plusieurs valeurs si elles ont été créées à l'aide de Gremlin. En d'autres termes, une propriété de nœud ou de relation peut avoir un ensemble de valeurs différentes plutôt qu'une seule. Neptune a étendu la sémantique d'openCypher pour gérer les propriétés à valeurs multiples sans problème.

Les types de données pris en charge sont documentés dans Format de données openCypher. Cependant, pour le moment, nous ne recommandons pas d'insérer des valeurs de propriétés `Array` dans un graphe openCypher. Bien qu'il soit possible d'insérer une valeur de propriété de tableau à l'aide du chargeur en bloc, la version actuelle de Neptune openCypher la traite comme un ensemble de propriétés à valeurs multiples plutôt que comme une valeur de liste unique.

Vous trouverez ci-dessous la liste des types de données pris en charge dans cette version :

- `Bool`
- `Byte`
- `Short`
- `Int`

- `Long`
- `Float` (inclut Infinity et NaN plus et moins, mais pas INF)
- `Double` (inclut Infinity et NaN plus et moins, mais pas INF)
- `DateTime`
- `String`

La fonctionnalité d'explication d'OpenCypher

La fonctionnalité `explain` openCypher est un outil en libre-service dans Amazon Neptune qui vous aide à comprendre l'approche d'exécution adoptée par le moteur Neptune. Pour invoquer explain, vous devez transmettre un paramètre à une requête HTTPS openCypher avec `explain=`*`mode`*, où la valeur *mode* peut être l'une des suivantes :

- **`static`** : en mode `static`, `explain` affiche uniquement la structure statique du plan de requête. Il n'exécute pas réellement la requête.
- **`dynamic`** : en mode `dynamic`, `explain` exécute également la requête et inclut les aspects dynamiques du plan de requête. Ces aspects peuvent inclure le nombre de liaisons intermédiaires transitant via les opérateurs et le ratio de liaisons sortantes par rapport aux liaisons entrantes, ainsi que le temps total pris par chaque opérateur.
- **`details`** : en mode `details`, `explain` imprime les informations affichées en mode dynamique, ainsi que des détails supplémentaires tels que la chaîne de requête openCypher réelle et le nombre de plages estimé pour le modèle sous-jacent d'un opérateur de jointure.

Par exemple, en utilisant `POST` le `dynamic` mode with :

AWS CLI

```
aws neptunedata execute-open-cypher-explain-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "MATCH (n) RETURN n LIMIT 1" \
  --explain-mode dynamic
```

Pour plus d'informations, consultez execute-open-cypher-explain-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_explain_query(
    openCypherQuery='MATCH (n) RETURN n LIMIT 1',
    explainMode='dynamic'
)

print(response['results'].read().decode('utf-8'))
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "query=MATCH (n) RETURN n LIMIT 1" \
  -d "explain=dynamic"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=MATCH (n) RETURN n LIMIT 1" \
  -d "explain=dynamic"
```

Explication des limites d'OpenCypher dans Neptune

La version actuelle d'openCypher explain présente les limites suivantes :

- Les plans explain ne sont actuellement disponibles que pour les requêtes qui effectuent des opérations en lecture seule. Les requêtes qui effectuent n'importe quel type de mutation, telles que `CREATE`, `DELETE`, `MERGE`, `SET` etc. ne sont pas prises en charge.
- Les opérateurs et les résultats d'un plan spécifique peuvent changer dans les versions ultérieures.

Les opérateurs DFE dans OpenCypher expliquent la sortie

Pour utiliser les informations fournies par la fonctionnalité openCypher `explain`, vous devez comprendre certains détails sur le fonctionnement du moteur de requêtes DFE (le DFE est le moteur utilisé par Neptune pour traiter les requêtes openCypher).

Le moteur DFE convertit chaque requête en un pipeline d'opérateurs. À partir du premier opérateur, des solutions intermédiaires circulent d'un opérateur au suivant dans ce pipeline d'opérateurs. Chaque ligne de la table explain représente un résultat, jusqu'au point d'évaluation.

Les opérateurs qui peuvent apparaître dans un plan de requête DFE sont les suivants :

DFEApply : exécute la fonction spécifiée dans la section des arguments, sur la valeur stockée dans la variable spécifiée

DFEBindRelation— Lie les variables portant les noms spécifiés

DFEChunkLocalSubQuery— Il s'agit d'une opération non bloquante qui enveloppe les sous-requêtes en cours d'exécution.

DFEDistinctColumn— Renvoie le sous-ensemble distinct des valeurs d'entrée en fonction de la variable spécifiée.

DFEDistinctRelation— Renvoie le sous-ensemble distinct des solutions d'entrée en fonction de la variable spécifiée.

DFEDrain : apparaît à la fin d'une sous-requête pour agir comme étape de résiliation de cette sous-requête. Le nombre de solutions est enregistré en tant qu'`Units In`. `Units Out` est toujours égal à zéro.

DFEForwardValue— Copie tous les fragments d'entrée directement en tant que fragments de sortie à transmettre à son opérateur en aval.

DFEGroupByHashIndex— Effectue une opération de regroupement sur les solutions d'entrée en fonction d'un indice de hachage précédemment calculé (à l'aide de l'`DFEHashIndexBuild`opération). En tant que sortie, l'entrée donnée est prolongée par une colonne contenant une clé de groupe pour chaque solution d'entrée.

DFEHashIndexBuild— Construit un index de hachage sur un ensemble de variables comme effet secondaire. Cet indice de hachage est généralement réutilisé dans les opérations ultérieures. Consultez `DFEHashIndexJoin` ou `DFEGroupByHashIndex` pour savoir où cet index de hachage peut être utilisé.

DFEHashIndexJoin— Effectue une jointure entre les solutions entrantes par rapport à un index de hachage créé précédemment. Consultez `DFEHashIndexBuild` pour savoir où cet index de hachage peut être créé.

DFEJoinExists— Prend une relation d'entrée gauche et droite, et conserve les valeurs de la relation de gauche qui ont une valeur correspondante dans la relation de droite telle que définie par les variables de jointure données.

 : opération non bloquante qui agit comme un wrapper pour une sous-requête, ce qui permet de l'exécuter à plusieurs reprises pour une utilisation en boucle.

DFEMergeChunks— Il s'agit d'une opération de blocage qui combine des segments provenant de son opérateur en amont en un seul bloc de solutions à transmettre à son opérateur en aval (inverse de`DFESplitChunks`).

DFEMinus : prend une relation d'entrée gauche et droite et conserve les valeurs de la relation de gauche qui ont une valeur correspondante dans la relation de droite telle que définie par les variables de jointure données. S'il n'y a aucun chevauchement entre les variables des deux relations, cet opérateur renvoie simplement une relation vide.

DFENotExists— Prend une relation d'entrée gauche et droite et conserve les valeurs de la relation de gauche qui n'ont pas de valeur correspondante dans la relation de droite telle que définie par les variables de jointure données. S'il n'y a aucun chevauchement entre les variables des deux relations, cet opérateur renvoie une relation vide.

DFEOptionalJoin— Effectue une jointure externe gauche (également appelée jointure FACULTATIVE) : les solutions du côté gauche qui ont au moins un partenaire de jointure sur le côté droit sont jointes, et les solutions du côté gauche sans partenaire de liaison sur le côté droit sont transmises telles quelles. Il s'agit d'une opération de blocage.

DFEPipelineJoin— Joint l'entrée au modèle de tuple défini par l'`pattern`argument.

DFEPipelineRangeCount— Compte le nombre de solutions correspondant à un modèle donné et renvoie une seule solution uniaire contenant la valeur du comptage.

DFEPipelineScan— Analyse la base de données à la recherche de l'`pattern`argument donné, avec ou sans filtre donné sur les colonnes.

DFEProject : utilise plusieurs colonnes d'entrée et ne projette que les colonnes souhaitées.

DFEReduce : exécute la fonction d'agrégation spécifiée au niveau des variables spécifiées.

DFERelationalJoin— Joint l'entrée de l'opérateur précédent en fonction des clés de modèle spécifiées à l'aide d'une jointure par fusion. Il s'agit d'une opération de blocage.

DFERouteChunks— Prélève les fragments d'entrée depuis son bord entrant unique et achemine ces morceaux le long de ses multiples arêtes sortantes.

DFESelectRows— Cet opérateur prend des lignes de manière sélective à partir de ses solutions de relation d'entrée de gauche pour les transmettre à son opérateur en aval. Les lignes sont sélectionnées en fonction des identifiants de ligne fournis dans la relation d'entrée appropriée de l'opérateur.

DFESerialize : sérialise les résultats finaux d'une requête sous forme de chaîne JSON, en mappant chaque solution d'entrée au nom de variable approprié. Pour les résultats relatifs aux nœuds et aux périphéries, ces résultats sont sérialisés dans une carte des propriétés et des métadonnées des entités.

DFESort : prend une relation d'entrée et produit une relation triée en fonction de la clé de tri fournie.

DFESplitByGroup— Divise chaque segment d'entrée d'un bord entrant en petits morceaux de sortie correspondant aux groupes de lignes identifiés par les identifiants de ligne du segment d'entrée correspondant de l'autre bord entrant.

DFESplitChunks— Divise chaque segment d'entrée en morceaux de sortie plus petits (inverse de). `DFEMergeChunks`

DFEStreamingHashIndexBuild— Version en streaming de`DFEHashIndexBuild`.

DFEStreamingGroupByHashIndex— Version en streaming de`DFEGroupByHashIndex`.

DFESubquery : cet opérateur apparaît au début de tous les plans et encapsule les parties du plan exécutées sur le moteur DFE, qui est le plan complet pour openCypher.

DFESymmetricHashJoin— Joint l'entrée de l'opérateur précédent en fonction des clés de modèle spécifiées à l'aide d'une jointure par hachage. Il s'agit d'une opération non bloquante.

DFESync : cet opérateur est un opérateur de synchronisation qui prend en charge les plans non bloquants. Il prend des solutions provenant de deux périphéries entrantes et les transmet aux périphéries aval appropriées. À des fins de synchronisation, les entrées situées le long de l'un de ces périphéries peuvent être mises en mémoire tampon en interne.

DFETee : opérateur de branchement qui envoie le même ensemble de solutions à plusieurs opérateurs.

DFETermResolution— Effectue une opération de localisation ou de globalisation sur ses entrées, ce qui donne lieu à des colonnes d'identifiants localisés ou globalisés respectivement.

: déplie les listes de valeurs d'une colonne d'entrée dans la colonne de sortie sous forme d'éléments individuels.

DFEUnion : prend au moins deux relations d'entrée et produit une union de ces relations en utilisant le schéma de sortie souhaité.

SolutionInjection— Apparaît avant tout le reste dans la sortie d'explication, avec une valeur de 1 dans la colonne Unités sorties. Cependant, il dessert une déclaration no-op et n'injecte aucune solution dans le moteur DFE.

TermResolution— Apparaît à la fin des plans et traduit les objets du moteur Neptune en objets OpenCypher.

Les colonnes d'OpenCypher expliquent le résultat

Les informations du plan de requête que Neptune génère sous forme de sortie openCypher explain contiennent des tables avec un opérateur par ligne. Cette table possède les colonnes suivantes :

ID : identifiant numérique de cet opérateur dans le plan.

Out #1 (et Out #2) : identifiant(s) des opérateur(s) qui se trouvent en aval de cet opérateur. Il peut y avoir au plus deux opérateurs en aval.

Nom : nom de cet opérateur.

Arguments : tous les détails pertinents concernant l'opérateur. Cela inclut des éléments tels que le schéma d'entrée, le schéma de sortie, le modèle (pour `PipelineScan` et `PipelineJoin`), etc.

Mode : étiquette décrivant le comportement fondamental de l'opérateur. Cette colonne est généralement vide (`-`). Une exception est `TermResolution`, où le mode peut être`id2value_opencypher`, indiquant une résolution entre l'ID et la valeur openCypher.

Unités en entrée : nombre de solutions transmises en entrée à cet opérateur. Les opérateurs sans opérateurs en amont, tels que `DFEPipelineScan`, `SolutionInjections` et `DFESubquery` sans valeur statique injectée, ont une valeur nulle.

Unités en sortie : nombre de solutions générées en sortie par cet opérateur. `DFEDrain` est un cas particulier, où le nombre de solutions drainées est enregistré dans `Units In`, et `Units Out` est toujours égal à zéro.

Ratio : ratio entre les éléments `Units Out` et `Units In`.

Temps (ms) : temps CPU consommé par cet opérateur, en millisecondes.

Exemple de base de la sortie openCypher explain

Vous trouverez ci-dessous un exemple de base de la sortie openCypher `explain`. La requête est une recherche à nœud unique dans le jeu de données des routes aériennes pour un nœud `ATL` dont le code d'aéroport est invoqué `explain` en utilisant le `details` mode au format de sortie ASCII par défaut.

`explain`Pour appeler cette requête, procédez comme suit :

AWS CLI

```
aws neptunedata execute-open-cypher-explain-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "MATCH (n {code: 'ATL'}) RETURN n" \
  --explain-mode details
```

Pour plus d'informations, consultez execute-open-cypher-explain-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
```

```
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_explain_query(
    openCypherQuery="MATCH (n {code: 'ATL'}) RETURN n",
    explainMode='details'
)

print(response['results'].read().decode('utf-8'))
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "query=MATCH (n {code: 'ATL'}) RETURN n" \
  -d "explain=details"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=MATCH (n {code: 'ATL'}) RETURN n" \
  -d "explain=details"
```

Le `explain` résultat :

```
Query:
MATCH (n {code: 'ATL'}) RETURN n

################################################################################################
```

```
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
#########################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 1         # 0.00  # 4.00      #
#########################################################################################
# 2  # -      # -      # TermResolution    # vars=[?n]          # id2value_opencypher #
 1        # 1         # 1.00  # 2.00      #
#########################################################################################


subQuery1
#########################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?n) with property 'code'
 as ?n_code2 and label 'ALL'                                       # -    # 0
   # 1         # 0.00  # 0.21      #
#    #        #        #                      # inlineFilters=[(?n_code2 IN
 ["ATL"^^xsd:string])]                                                  #
  #          #           #       #           #
#    #        #        #                        # patternEstimate=1
                                                                 #      #
  #           #       #           #
#########################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#9d84f97c-c3b0-459a-98d5-955a8726b159/graph_1 # -    #
 1        # 1         # 1.00  # 0.04      #
#########################################################################################
# 2  # 3      # -      # DFEProject            # columns=[?n]
                                                                 # -    # 1
  # 1         # 1.00  # 0.04      #
#########################################################################################
# 3  # -      # -      # DFEDrain              # -
                                                                 # -    # 1
  # 0         # 0.00  # 0.03      #
#########################################################################################
```

```
subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#9d84f97c-
c3b0-459a-98d5-955a8726b159/graph_1
##################################################################################################
# ID # Out #1 # Out #2 # Name                 # Arguments
                       # Mode # Units In # Units Out # Ratio # Time (ms) #
##################################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?n, ?n_code2]
                       # -    # 0        # 1          # 0.00  # 0.02      #
##################################################################################################
# 1  # 2      # 3      # DFETee               # -
                       # -    # 1        # 2          # 2.00  # 0.02      #
##################################################################################################
# 2  # 4      # -      # DFEDistinctColumn    # column=?n
                       # -    # 1        # 1          # 1.00  # 0.20      #
#    #        #        #                      # ordered=false
                       #      #          #           #       #           #
##################################################################################################
# 3  # 5      # -      # DFEHashIndexBuild    # vars=[?n]
                       # -    # 1        # 1          # 1.00  # 0.04      #
##################################################################################################
# 4  # 5      # -      # DFEPipelineJoin      # pattern=Node(?n) with property 'ALL'
 and label '?n_label1' # -    # 1        # 1          # 1.00  # 0.25      #
#    #        #        #                      # patternEstimate=3506
                       #      #          #           #       #           #
##################################################################################################
# 5  # 6      # 7      # DFESync              # -
                       # -    # 2        # 2          # 1.00  # 0.02      #
##################################################################################################
# 6  # 8      # -      # DFEForwardValue      # -
                       # -    # 1        # 1          # 1.00  # 0.01      #
##################################################################################################
# 7  # 8      # -      # DFEForwardValue      # -
                       # -    # 1        # 1          # 1.00  # 0.01      #
##################################################################################################
# 8  # 9      # -      # DFEHashIndexJoin     # -
                       # -    # 2        # 1          # 0.50  # 0.35      #
##################################################################################################
# 9  # -      # -      # DFEDrain             # -
                       # -    # 1        # 0          # 0.00  # 0.02      #
##################################################################################################
```

Au niveau supérieur, `SolutionInjection` apparaît avant tout le reste, avec une unité en sortie. Notez qu'il n'injecte aucune solution. Vous pouvez voir que l'opérateur suivant, `DFESubquery`, n'a aucune unité en entrée.

Après `SolutionInjection` au niveau supérieur, figurent les opérateurs `DFESubquery` et `TermResolution`. `DFESubquery` encapsule les parties du plan d'exécution des requêtes qui sont transmises au moteur DFE (pour les requêtes openCypher, le plan de requête complet est exécuté par le DFE). Tous les opérateurs du plan de requête sont imbriqués dans `subQuery1` qui est référencé par `DFESubquery`. La seule exception est `TermResolution`, qui matérialise les ID internes en objets openCypher entièrement sérialisés.

Tous les opérateurs redirigés vers le moteur DFE ont des noms qui commencent par un préfixe `DFE`. Comme mentionné ci-dessus, l'ensemble du plan de requête openCypher est exécuté par le DFE. Par conséquent, tous les opérateurs, à l'exception de l'opérateur `TermResolution` final, commencent par `DFE`.

Dans `subQuery1`, il peut y avoir zéro opérateur `DFEChunkLocalSubQuery` ou `DFELoopSubQuery` ou plus encapsulant une partie du plan d'exécution transmis qui est exécuté dans un mécanisme limité à la mémoire. `DFEChunkLocalSubQuery` contient ici un seul élément `SolutionInjection` qui est utilisé comme entrée pour la sous-requête. Pour trouver la table correspondant à cette sous-requête dans la sortie, recherchez l'`subQuery=`*`graph URI`* spécifié dans la colonne `Arguments` pour l'opérateur `DFEChunkLocalSubQuery` ou `DFELoopSubQuery`.

Dans `subQuery1`, `DFEPipelineScan` avec l'`ID` 0 analyse la base de données à la recherche d'un modèle (`pattern`) spécifié. Le modèle recherche une entité dont la propriété `code` est enregistrée sous forme de variable `?n_code2` sur toutes les étiquettes (vous pouvez filtrer une étiquette spécifique en ajoutant `airport` à `n:airport`). L'argument `inlineFilters` indique que le filtrage de la propriété `code` est égal à `ATL`.

Ensuite, l'opérateur `DFEChunkLocalSubQuery` joint les résultats intermédiaires d'une sous-requête contenant `DFEPipelineJoin`. Cela garantit que `?n` est bien un nœud, puisque le l'opération `DFEPipelineScan` précédente analyse toute entité possédant la propriété `code`.

Exemple de sortie d'**`explication`** pour une recherche de relation avec une limite

Cette requête recherche les relations entre deux nœuds anonymes de type `route` et en renvoie au maximum 10. Encore une fois, le mode `explain` est `details`, et le format ASCII est le format de sortie par défaut.

Ici, `DFEPipelineScan` recherche les arêtes qui commencent par un nœud anonyme `?anon_node7` et se terminent par un autre nœud anonyme `?anon_node21`, avec un type de relation enregistré sous la forme `?p_type1`. Il existe un filtre pour les éléments `?p_type1` qui correspondent à `el://route` (où `el` désigne l'étiquette d'une arête), ce qui équivaut à `[p:route]` dans la chaîne de requête.

`DFEDrain` collecte la solution de sortie avec une limite de 10, comme indiqué dans sa colonne `Arguments`. `DFEDrain` prend fin une fois que la limite est atteinte ou que toutes les solutions sont générées, selon la situation qui survient en premier.

`explain`Pour appeler cette requête, procédez comme suit :

AWS CLI

```
aws neptunedata execute-open-cypher-explain-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "MATCH ()-[p:route]->() RETURN p LIMIT 10" \
  --explain-mode details
```

Pour plus d'informations, consultez execute-open-cypher-explain-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_explain_query(
    openCypherQuery='MATCH ()-[p:route]->() RETURN p LIMIT 10',
    explainMode='details'
)

print(response['results'].read().decode('utf-8'))
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "query=MATCH ()-[p:route]->() RETURN p LIMIT 10" \
  -d "explain=details"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=MATCH ()-[p:route]->() RETURN p LIMIT 10" \
  -d "explain=details"
```

Le `explain` résultat :

```
Query:
MATCH ()-[p:route]->() RETURN p LIMIT 10

#################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
#################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 10        # 0.00  # 5.00      #
#################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?p]          # id2value_opencypher #
 10       # 10        # 1.00  # 1.00      #
#################################################################################################
```

```
subQuery1
#######################################################################################################
# ID # Out #1 # Out #2 # Name            # Arguments
              # Mode # Units In # Units Out # Ratio # Time (ms) #
#######################################################################################################
# 0  # 1      # -      # DFEPipelineScan # pattern=Edge((?anon_node7)-[?p:?p_type1]->(?
anon_node21)) # -    # 0        # 1000      # 0.00  # 0.66      #
#    #        #        #                 # inlineFilters=[(?p_type1 IN [<el://route>])]
              #      #          #           #       #           #
#    #        #        #                 # patternEstimate=26219
              #      #          #           #       #           #
#######################################################################################################
# 1  # 2      # -      # DFEProject      # columns=[?p]
              # -    # 1000     # 1000      # 1.00  # 0.14      #
#######################################################################################################
# 2  # -      # -      # DFEDrain        # limit=10
              # -    # 1000     # 0         # 0.00  # 0.11      #
#######################################################################################################
```

Exemple de sortie d'**`explication`** pour une fonction d'expression de valeur

La fonction est :

```
MATCH (a) RETURN DISTINCT labels(a)
```

Dans la sortie `explain` ci-dessous, `DFEPipelineScan` (ID 0) recherche toutes les étiquettes des nœuds. Cela correspond à `MATCH (a`.

`DFEChunkLocalSubquery` (ID 1) agrège l'étiquette de `?a` pour chaque `?a`. Cela correspond à `labels(a)`. Vous pouvez le voir via `DFEApply` et `DFEReduce`.

`BindRelation` (ID 2) est utilisé pour remplacer le nom de la valeur générique de colonne `?__gen_labelsOfa2` par `?labels(a)`.

`DFEDistinctRelation` (ID 4) récupère uniquement les étiquettes distinctes (plusieurs nœuds :airport donneraient des étiquettes (a): ["airport"] dupliquées). Cela correspond à `DISTINCT labels(a)`.

`explain`Pour appeler cette requête, procédez comme suit :

AWS CLI

```
aws neptunedata execute-open-cypher-explain-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "MATCH (a) RETURN DISTINCT labels(a)" \
  --explain-mode details
```

Pour plus d'informations, consultez execute-open-cypher-explain-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_explain_query(
    openCypherQuery='MATCH (a) RETURN DISTINCT labels(a)',
    explainMode='details'
)

print(response['results'].read().decode('utf-8'))
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "query=MATCH (a) RETURN DISTINCT labels(a)" \
  -d "explain=details"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=MATCH (a) RETURN DISTINCT labels(a)" \
  -d "explain=details"
```

Le `explain` résultat :

```
Query:
MATCH (a) RETURN DISTINCT labels(a)

############################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
############################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
############################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 5         # 0.00  # 81.00     #
############################################################################################
# 2  # -      # -      # TermResolution    # vars=[?labels(a)]  # id2value_opencypher #
 5        # 5         # 1.00  # 1.00      #
############################################################################################


subQuery1
############################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
############################################################################################
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?a) with property 'ALL'
 and label '?a_label1'                                             # -    # 0
    # 3750      # 0.00  # 26.77     #
```

```
#    #        #        #                       # patternEstimate=3506
                                                                    #      #
  #           #       #           #
#########################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#8b314f55-2cc7-456a-a48a-c76a0465cfab/graph_1 # -    #
 3750     # 3750      # 1.00  # 0.04      #
#########################################################################################
# 2  # 3      # -      # DFEBindRelation       # inputVars=[?a, ?__gen_labelsOfa2, ?
__gen_labelsOfa2]                                                    # -    # 3750
     # 3750      # 1.00  # 0.08      #
#    #        #        #                       # outputVars=[?a, ?__gen_labelsOfa2, ?
labels(a)]                                                           #      #
    #           #       #           #
#########################################################################################
# 3  # 4      # -      # DFEProject            # columns=[?labels(a)]
                                                                    # -    # 3750
  # 3750      # 1.00  # 0.05      #
#########################################################################################
# 4  # 5      # -      # DFEDistinctRelation   # -
                                                                    # -    # 3750
  # 5         # 0.00  # 2.78      #
#########################################################################################
# 5  # -      # -      # DFEDrain              # -
                                                                    # -    # 5
  # 0         # 0.00  # 0.03      #
#########################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#8b314f55-2cc7-456a-
a48a-c76a0465cfab/graph_1
#########################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                    # Mode     # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?a]
                    # -        # 0        # 3750      # 0.00  # 0.02      #
#########################################################################################
# 1  # 2      # 3      # DFETee               # -
                    # -        # 3750     # 7500      # 2.00  # 0.02      #
#########################################################################################
# 2  # 4      # -      # DFEProject           # columns=[?a]
                    # -        # 3750     # 3750      # 1.00  # 0.04      #
#########################################################################################
```

```
# 3  # 17     # -      # DFEOptionalJoin      # -
                    # -        # 7500     # 3750      # 0.50  # 0.44      #
##################################################################################################
# 4  # 5      # -      # DFEDistinctRelation  # -
                    # -        # 3750     # 3750      # 1.00  # 2.23      #
##################################################################################################
# 5  # 6      # -      # DFEDistinctColumn    # column=?a
                    # -        # 3750     # 3750      # 1.00  # 1.50      #
#    #        #        #                      # ordered=false
                    #          #          #           #       #           #
##################################################################################################
# 6  # 7      # -      # DFEPipelineJoin      # pattern=Node(?a) with property 'ALL'
 and label '?a_label3' # -        # 3750     # 3750      # 1.00  # 10.58     #
#    #        #        #                      # patternEstimate=3506
                    #          #          #           #       #           #
##################################################################################################
# 7  # 8      # 9      # DFETee               # -
                    # -        # 3750     # 7500      # 2.00  # 0.02      #
##################################################################################################
# 8  # 10     # -      # DFEBindRelation      # inputVars=[?a_label3]
                    # -        # 3750     # 3750      # 1.00  # 0.04      #
#    #        #        #                      # outputVars=[?100]
                    #          #          #           #       #           #
##################################################################################################
# 9  # 11     # -      # DFEBindRelation      # inputVars=[?a, ?a_label3, ?100]
                    # -        # 7500     # 3750      # 0.50  # 0.07      #
#    #        #        #                      # outputVars=[?a, ?a_label3, ?100]
                    #          #          #           #       #           #
##################################################################################################
# 10 # 9      # -      # DFETermResolution    # column=?100
                    # id2value # 3750     # 3750      # 1.00  # 7.60      #
##################################################################################################
# 11 # 12     # -      # DFEBindRelation      # inputVars=[?a, ?a_label3, ?100]
                    # -        # 3750     # 3750      # 1.00  # 0.06      #
#    #        #        #                      # outputVars=[?a, ?100, ?a_label3]
                    #          #          #           #       #           #
##################################################################################################
# 12 # 13     # -      # DFEApply             # functor=nodeLabel(?a_label3)
                    # -        # 3750     # 3750      # 1.00  # 0.55      #
##################################################################################################
# 13 # 14     # -      # DFEProject           # columns=[?a, ?a_label3_alias4]
                    # -        # 3750     # 3750      # 1.00  # 0.05      #
##################################################################################################
```

```
# 14 # 15     # -      # DFEMergeChunks       # -
                    # -        # 3750     # 3750      # 1.00  # 0.02      #
#########################################################################################
# 15 # 16     # -      # DFEReduce            # functor=collect(?a_label3_alias4)
                    # -        # 3750     # 3750      # 1.00  # 6.37      #
#    #        #        #                     # segmentationKey=[?a]
                    #          #          #           #       #           #
#########################################################################################
# 16 # 3      # -      # DFEMergeChunks       # -
                    # -        # 3750     # 3750      # 1.00  # 0.03      #
#########################################################################################
# 17 # -      # -      # DFEDrain             # -
                    # -        # 3750     # 0         # 0.00  # 0.02      #
#########################################################################################
```

Exemple de sortie d'**`explication`** pour une fonction d'expression de valeur mathématique

Dans cet exemple, `RETURN abs(-10)` effectue une évaluation simple en prenant la valeur absolue d'une constante, `-10`.

`DFEChunkLocalSubQuery` (ID 1) effectue une injection de solution pour la valeur statique `-10`, qui est stockée dans la variable `?100`.

`DFEApply` (ID 2) est l'opérateur qui exécute la fonction de valeur absolue `abs()` au niveau de la valeur statique stockée dans la variable `?100`.

Voici la requête et le `explain` résultat obtenu.

`explain`Pour appeler cette requête, procédez comme suit :

AWS CLI

```
aws neptunedata execute-open-cypher-explain-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "RETURN abs(-10)" \
  --explain-mode details
```

Pour plus d'informations, consultez execute-open-cypher-explain-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_explain_query(
    openCypherQuery='RETURN abs(-10)',
    explainMode='details'
)

print(response['results'].read().decode('utf-8'))
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "query=RETURN abs(-10)" \
  -d "explain=details"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=RETURN abs(-10)" \
  -d "explain=details"
```

Le `explain` résultat :

```
Query:
RETURN abs(-10)

################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments             # Mode
  # Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]        # -
  # 0        # 1        # 0.00  # 0         #
################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1    # -
  # 0        # 1        # 0.00  # 4.00      #
################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?_internalVar1] #
 id2value_opencypher # 1        # 1         # 1.00  # 1.00      #
################################################################################################


subQuery1
################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFESolutionInjection  # outSchema=[]
                                                                 # -    # 0
  # 1         # 0.00  # 0.01      #
################################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#c4cc6148-cce3-4561-93c0-deb91f257356/graph_1 # -    #
 1        # 1         # 1.00  # 0.03      #
################################################################################################
# 2  # 3      # -      # DFEApply             # functor=abs(?100)
                                                                 # -    # 1
  # 1         # 1.00  # 0.26      #
################################################################################################
# 3  # 4      # -      # DFEBindRelation       # inputVars=[?_internalVar2, ?
_internalVar2]                                                                  # -
 # 1        # 1         # 1.00  # 0.04      #
#    #        #        #                      # outputVars=[?_internalVar2, ?
_internalVar1]                                                                 #
 #          #           #       #           #
```

```
#########################################################################################
# 4  # 5      # -      # DFEProject            # columns=[?_internalVar1]
                                                                    # -    # 1
  # 1         # 1.00  # 0.06      #
#########################################################################################
# 5  # -      # -      # DFEDrain              # -
                                                                    # -    # 1
  # 0         # 0.00  # 0.05      #
#########################################################################################

subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#c4cc6148-
cce3-4561-93c0-deb91f257356/graph_1
#########################################################################################
# ID # Out #1 # Out #2 # Name                 # Arguments                          #
 Mode # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFESolutionInjection # solutions=[?100 -> [-10^^<LONG>]]   # -
    # 0        # 1         # 0.00  # 0.01      #
#    #        #        #                      # outSchema=[?100]                   #
    #          #           #       #           #
#########################################################################################
# 1  # 3      # -      # DFERelationalJoin    # joinVars=[]                        # -
    # 2        # 1         # 0.50  # 0.18      #
#########################################################################################
# 2  # 1      # -      # DFESolutionInjection # outSchema=[]                      # -
    # 0        # 1         # 0.00  # 0.01      #
#########################################################################################
# 3  # -      # -      # DFEDrain              # -                                  # -
    # 1        # 0         # 0.00  # 0.02      #
#########################################################################################
```

Exemple de sortie d'**`explication`** pour une requête de chemin de longueur variable (VLP)

Il s'agit d'un exemple de plan de requête plus complexe pour gérer une requête de chemin de longueur variable. Pour plus de clarté, cet exemple ne montre qu'une partie de la sortie `explain`.

Dans `subQuery1`, `DFEPipelineScan` (ID 0) et `DFEChunkLocalSubQuery` (ID 1), qui injectent la sous-requête `...graph_1`, sont chargés de rechercher un nœud contenant le code `YP0`.

Dans `subQuery1`, `DFEChunkLocalSubQuery` (ID 2), qui injecte la sous-requête `...graph_2`, est chargé de rechercher un nœud contenant le code `LAX`.

Dans `subQuery1`, `DFEChunkLocalSubQuery` (ID 3) injecte la sous-requête `...graph3`, qui contient `DFELoopSubQuery` (ID 17), qui à son tour injecte la sous-requête `...graph5`. Cette opération est chargée de résoudre le modèle de longueur variable `-[*2]->` dans la chaîne de requête entre deux nœuds.

`explain`Pour appeler cette requête, procédez comme suit :

AWS CLI

```
aws neptunedata execute-open-cypher-explain-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p" \
  --explain-mode details
```

Pour plus d'informations, consultez execute-open-cypher-explain-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_explain_query(
    openCypherQuery="MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p",
    explainMode='details'
)

print(response['results'].read().decode('utf-8'))
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
```

```
  -d "query=MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p" \
  -d "explain=details"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p" \
  -d "explain=details"
```

Le `explain` résultat :

```
Query:
MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p

################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 0         # 0.00  # 84.00     #
################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?p]          # id2value_opencypher #
 0        # 0         # 0.00  # 0         #
################################################################################################


subQuery1
################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
################################################################################################
```

```
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?a) with property 'code'
 as ?a_code7 and label 'ALL'                                          # -    # 0
   # 1         # 0.00  # 0.68      #
#    #        #        #                       # inlineFilters=[(?a_code7 IN
 ["YPO"^^xsd:string])]                                                    #
  #          #           #       #           #
#    #        #        #                       # patternEstimate=1
                                                                      #      #
  #          #       #           #
#############################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_1 # -    #
 1        # 1         # 1.00  # 0.03      #
#############################################################################################
# 2  # 3      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_2 # -    #
 1        # 1         # 1.00  # 0.02      #
#############################################################################################
# 3  # 4      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_3 # -    #
 1        # 0         # 0.00  # 0.04      #
#############################################################################################
# 4  # 5      # -      # DFEBindRelation       # inputVars=[?__gen_path6, ?
anon_rel26, ?b_code8, ?b, ?a_code7, ?a, ?__gen_path6]                          # -
   # 0        # 0         # 0.00  # 0.10      #
#    #        #        #                       # outputVars=[?__gen_path6, ?
anon_rel26, ?b_code8, ?b, ?a_code7, ?a, ?p]                                    #
  #          #           #       #           #
#############################################################################################
# 5  # 6      # -      # DFEProject            # columns=[?p]
                                                                      # -    # 0
  # 0         # 0.00  # 0.05      #
#############################################################################################
# 6  # -      # -      # DFEDrain              # -
                                                                      # -    # 0
  # 0         # 0.00  # 0.02      #
#############################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_1
#############################################################################################
# ID # Out #1 # Out #2 # Name                 # Arguments
                    # Mode # Units In # Units Out # Ratio # Time (ms) #
```

```
#########################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?a, ?a_code7]
                   # -    # 0        # 1         # 0.00  # 0.01      #
#########################################################################################
# 1  # 2      # 3      # DFETee               # -
                   # -    # 1        # 2         # 2.00  # 0.01      #
#########################################################################################
# 2  # 4      # -      # DFEDistinctColumn    # column=?a
                   # -    # 1        # 1         # 1.00  # 0.25      #
#    #        #        #                      # ordered=false
                   #      #          #           #       #           #
#########################################################################################
# 3  # 5      # -      # DFEHashIndexBuild    # vars=[?a]
                   # -    # 1        # 1         # 1.00  # 0.05      #
#########################################################################################
# 4  # 5      # -      # DFEPipelineJoin      # pattern=Node(?a) with property 'ALL'
 and label '?a_label1' # -    # 1        # 1         # 1.00  # 0.47      #
#    #        #        #                      # patternEstimate=3506
                   #      #          #           #       #           #
#########################################################################################
# 5  # 6      # 7      # DFESync              # -
                   # -    # 2        # 2         # 1.00  # 0.04      #
#########################################################################################
# 6  # 8      # -      # DFEForwardValue      # -
                   # -    # 1        # 1         # 1.00  # 0.01      #
#########################################################################################
# 7  # 8      # -      # DFEForwardValue      # -
                   # -    # 1        # 1         # 1.00  # 0.01      #
#########################################################################################
# 8  # 9      # -      # DFEHashIndexJoin     # -
                   # -    # 2        # 1         # 0.50  # 0.26      #
#########################################################################################
# 9  # -      # -      # DFEDrain             # -
                   # -    # 1        # 0         # 0.00  # 0.02      #
#########################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_2
#########################################################################################
# ID # Out #1 # Out #2 # Name                # Arguments
                         # Mode # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
```

```
# 0  # 1      # -      # DFEPipelineScan      # pattern=Node(?b) with property 'code'
 as ?b_code8 and label 'ALL' # -    # 0        # 1         # 0.00  # 0.38      #
#    #        #        #                     # inlineFilters=[(?b_code8 IN
 ["LAX"^^xsd:string])]                #      #          #           #       #
  #
#    #        #        #                     # patternEstimate=1
                           #      #          #           #       #           #
#########################################################################################
# 1  # 2      # -      # DFEMergeChunks       # -
                           # -    # 1        # 1         # 1.00  # 0.02      #
#########################################################################################
# 2  # 4      # -      # DFERelationalJoin    # joinVars=[]
                           # -    # 2        # 1         # 0.50  # 0.19      #
#########################################################################################
# 3  # 2      # -      # DFESolutionInjection # outSchema=[?a, ?a_code7]
                           # -    # 0        # 1         # 0.00  # 0         #
#########################################################################################
# 4  # -      # -      # DFEDrain             # -
                           # -    # 1        # 0         # 0.00  # 0.01      #
#########################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_3
#########################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                        # Mode     #
 Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
...
# 17 # 18     # -      # DFELoopSubQuery       # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_5 # -
  # 1        # 2         # 2.00  # 0.31      #
...
```

Transactions dans Neptune openCypher

L'implémentation openCypher dans Amazon Neptune utilise la sémantique des transactions définie par Neptune. Cependant, les niveaux d'isolement fournis par le pilote Bolt ont des implications spécifiques sur la sémantique des transactions Bolt, comme décrit dans les sections ci-dessous.

Read-only Requêtes relatives aux transactions Bolt

Les requêtes en lecture seule peuvent être traitées de différentes manières, avec différents modèles de transaction et niveaux d'isolement, comme suit :

Requêtes de transaction implicites en lecture seule

Voici un exemple de transaction implicite en lecture seule :

```
public void executeReadImplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
          Config.builder().withEncryption()
                          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                          .build());

  // create the session config
  SessionConfig sessionConfig = SessionConfig.builder()
                                             .withFetchSize(1000)
                                             .withDefaultAccessMode(AccessMode.READ)
                                             .build();

  // run the query as access mode read
  driver.session(sessionConfig).readTransaction(new TransactionWork<String>()
    {
      final StringBuilder resultCollector = new StringBuilder();

      @Override
      public String execute(final Transaction tx)
      {
        // execute the query
        Result queryResult = tx.run(READ_QUERY);

        // Read the result
        for (Record record : queryResult.list())
        {
```

```
          for (String key : record.keys())
          {
            resultCollector.append(key)
                           .append(":")
                           .append(record.get(key).asNode().toString());
          }
        }
        return resultCollector.toString();
      }

    }
  );

  // close the driver.
  driver.close();
}
```

Comme les réplicas en lecture n'acceptent que les requêtes en lecture seule, toutes les requêtes portant sur ces réplicas s'exécutent sous forme de transactions implicites en lecture, quel que soit le mode d'accès défini dans la configuration de session. Neptune évalue les transactions implicites en lecture comme des requêtes en lecture seule selon la sémantique d'isolement SNAPSHOT.

En cas d'échec, les transactions implicites en lecture font par défaut l'objet d'une nouvelle tentative.

Requêtes de transaction de validation automatique en lecture seule

Voici un exemple de transaction de validation automatique en lecture seule :

```
public void executeAutoCommitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";

  // Create the session config.
  final SessionConfig sessionConfig = SessionConfig
    .builder()
    .withFetchSize(1000)
    .withDefaultAccessMode(AccessMode.READ)
    .build();
```

```
  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
    Config.builder()
          .withEncryption()
          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
          .build());

  // result collector
  final StringBuilder resultCollector = new StringBuilder();

  // create a session
  final Session session = driver.session(sessionConfig);

  // run the query
  final Result queryResult = session.run(READ_QUERY);
  for (final Record record : queryResult.list())
  {
    for (String key : record.keys())
    {
      resultCollector.append(key)
                     .append(":")
                     .append(record.get(key).asNode().toString());
    }
  }

  // close the session
  session.close();

  // close the driver
  driver.close();
}
```

Si le mode d'accès est défini sur `READ` dans la configuration de session, Neptune évalue les requêtes de transaction de validation automatique comme des requêtes en lecture seule selon une sémantique d'isolement `SNAPSHOT`. Notez que les réplicas en lecture n'acceptent que les requêtes en lecture seule.

Si vous ne transmettez pas de configuration de session, les requêtes de validation automatique sont traitées par défaut avec un isolement des requêtes de mutation. Il est donc important de transmettre une configuration de session qui définit explicitement le mode d'accès sur `READ`.

En cas d'échec, les requêtes de validation automatique en lecture seule ne font pas l'objet d'une nouvelle tentative.

Requêtes de transaction explicites en lecture seule

Voici un exemple de transaction explicite en lecture seule :

```
public void executeReadExplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";

  // Create the session config.
  final SessionConfig sessionConfig = SessionConfig
    .builder()
    .withFetchSize(1000)
    .withDefaultAccessMode(AccessMode.READ)
    .build();

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
    Config.builder()
          .withEncryption()
          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
          .build());

  // result collector
  final StringBuilder resultCollector = new StringBuilder();

  // create a session
  final Session session = driver.session(sessionConfig);

  // begin transaction
  final Transaction tx = session.beginTransaction();

  // run the query on transaction
  final List<Record> list = tx.run(READ_QUERY).list();

  // read the result
  for (final Record record : list)
  {
    for (String key : record.keys())
    {
      resultCollector
```

```
        .append(key)
        .append(":")
        .append(record.get(key).asNode().toString());
    }
  }

  // commit the transaction and for rollback we can use beginTransaction.rollback();
  tx.commit();

  // close the driver
  driver.close();
}
```

Si le mode d'accès est défini sur READ dans la configuration de session, Neptune évalue les transactions explicites en lecture seule comme des requêtes en lecture seule selon une sémantique d'isolement SNAPSHOT. Notez que les réplicas en lecture n'acceptent que les requêtes en lecture seule.

Si vous ne transmettez pas de configuration de session, les transactions explicites en lecture seule sont traitées par défaut avec un isolement des requêtes de mutation. Il est donc important de transmettre une configuration de session qui définit explicitement le mode d'accès sur READ.

En cas d'échec, les requêtes explicites en lecture seule font par défaut l'objet d'une nouvelle tentative.

Requêtes de transaction de mutation Bolt

Comme pour les requêtes en lecture seule, les requêtes de mutation peuvent être traitées de différentes manières, avec différents modèles de transaction et niveaux d'isolement, comme suit :

Requêtes de transaction de mutation implicites

Voici un exemple de transaction de mutation implicite :

```
public void executeWriteImplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // create node with label as label and properties.
  final String WRITE_QUERY = "CREATE (n:label {name : 'foo'})";

  // Read the vertex created with label as label.
```

```
final String READ_QUERY = "MATCH (n:label) RETURN n";

// create the driver
final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
  Config.builder()
        .withEncryption()
        .withTrustStrategy(TrustStrategy.trustSystemCertificates())
        .build());

// create the session config
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();

final StringBuilder resultCollector = new StringBuilder();

// run the query as access mode write
driver.session(sessionConfig).writeTransaction(new TransactionWork<String>()
{
  @Override
  public String execute(final Transaction tx)
  {
    // execute the write query and consume the result.
    tx.run(WRITE_QUERY).consume();

    // read the vertex written in the same transaction
    final List<Record> list = tx.run(READ_QUERY).list();

    // read the result
    for (final Record record : list)
    {
      for (String key : record.keys())
      {
        resultCollector
          .append(key)
          .append(":")
          .append(record.get(key).asNode().toString());
      }
    }
    return resultCollector.toString();
  }
}); // at the end, the transaction is automatically committed.
```

```
  // close the driver.
  driver.close();
}
```

Les lectures effectuées dans le cadre des requêtes de mutation sont exécutées avec un isolement `READ COMMITTED` avec les garanties habituelles applicables aux transactions de mutation Neptune.

Que vous transmettiez spécifiquement ou non une configuration de session, la transaction est toujours traitée comme une transaction d'écriture.

Pour les conflits, voir Résolution des conflits à l'aide de Lock-Wait délais.

Requêtes de transaction de mutation à validation automatique

Les requêtes de validation automatique des mutations héritent du même comportement que les transactions implicites de mutation.

Si vous ne transmettez pas de configuration de session, la transaction est traitée comme une transaction d'écriture par défaut.

En cas d'échec, les requêtes de validation automatique des mutations ne font pas automatiquement l'objet de nouvelles tentatives.

Requêtes de transaction de mutation explicites

Voici un exemple de transaction de mutation explicite :

```
public void executeWriteExplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // create node with label as label and properties.
  final String WRITE_QUERY = "CREATE (n:label {name : 'foo'})";

  // Read the vertex created with label as label.
  final String READ_QUERY = "MATCH (n:label) RETURN n";

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
    Config.builder()
          .withEncryption()
          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
```

```
          .build());

  // create the session config
  SessionConfig sessionConfig = SessionConfig
    .builder()
    .withFetchSize(1000)
    .withDefaultAccessMode(AccessMode.WRITE)
    .build();

  final StringBuilder resultCollector = new StringBuilder();

  final Session session = driver.session(sessionConfig);

  // run the query as access mode write
  final Transaction tx = driver.session(sessionConfig).beginTransaction();

  // execute the write query and consume the result.
  tx.run(WRITE_QUERY).consume();

  // read the result from the previous write query in a same transaction.
  final List<Record> list = tx.run(READ_QUERY).list();

  // read the result
  for (final Record record : list)
  {
    for (String key : record.keys())
    {
      resultCollector
        .append(key)
        .append(":")
        .append(record.get(key).asNode().toString());
    }
  }

  // commit the transaction and for rollback we can use tx.rollback();
  tx.commit();

  // close the session
  session.close();

  // close the driver.
  driver.close();
}
```

Les requêtes de mutation explicites héritent du même comportement que les transactions de mutation implicites.

Si vous ne transmettez pas de configuration de session, la transaction est traitée comme une transaction d'écriture par défaut.

Pour les conflits, voir Résolution des conflits à l'aide de Lock-Wait délais.

Streaming des résultats de requêtes avec le protocole Bolt

Lorsque vous exécutez une requête OpenCypher qui renvoie un grand nombre d'enregistrements, il n'est pas nécessaire de charger l'ensemble des résultats dans la mémoire du client en une seule fois. Neptune prend en charge le streaming des résultats via le protocole Bolt, qui permet au pilote de récupérer les enregistrements par lots et de les traiter progressivement au fur et à mesure qu'ils arrivent. Cela peut être avantageux si vous souhaitez traiter les résultats au fur et à mesure qu'ils sont renvoyés par le serveur, mais cela nécessite de configurer `FetchSize` et d'utiliser une itération différée pour éviter de charger le jeu de résultats complet en mémoire.

Cela fonctionne grâce au mécanisme BOLT v4.0 PULL. Au lieu de demander tous les enregistrements dans un seul message, le pilote envoie une demande PULL avec une taille de lot. Par exemple, « donnez-moi les 50 prochains enregistrements ». Neptune renvoie ce nombre d'enregistrements avec un indicateur indiquant si d'autres sont disponibles. Le pilote demande ensuite le lot suivant lorsque le client est prêt, et cela continue jusqu'à ce que le jeu de résultats soit entièrement consommé.

Vous pouvez contrôler la taille du lot à l'aide des `FetchSize` paramètres définis dans la configuration de session du pilote. Combiné à une itération lente (traitement des enregistrements un par un au fur et à mesure qu'ils sont renvoyés), cela limite l'utilisation de la mémoire côté client, quelle que soit la taille du jeu de résultats total.

> **Important**
>
> `FetchSize`contrôle le flux de données entre le serveur et le client. Cela n'a aucune incidence sur la façon dont Neptune exécute la requête. Neptune commence à exécuter la requête et à produire des résultats dès qu'il reçoit le message RUN. Les résultats sont mis en mémoire tampon côté serveur jusqu'à ce que le client les demande avec un message PULL. Si le client ne consomme pas les résultats dans le délai de requête configuré, Neptune met fin à la requête, supprime les résultats mis en mémoire tampon et libère les ressources côté serveur. Il n'y a pas de limite de mémoire tampon distincte côté serveur ; la mise en mémoire

tampon continue jusqu'à ce que le client extrait les résultats ou que le délai d'expiration de la requête soit atteint.
Le streaming est particulièrement efficace avec des requêtes non agrégantes qui peuvent produire des résultats de manière incrémentielle. Requêtes qui incluent des `ORDER BY` fonctions d'agrégation (telles que `count()collect()`, ou`sum()`) ou qui `DISTINCT` nécessitent que Neptune calcule le jeu de résultats complet avant de renvoyer des enregistrements. Dans ces cas, l'utilisation de la mémoire côté client par lot est `FetchSize` toujours limitée, mais le serveur doit conserver l'ensemble des résultats en mémoire avant le début de la diffusion.

> **Note**
>
> Le point de terminaison HTTPS OpenCypher fournit des résultats à l'aide du codage de transfert HTTP fragmenté, qui diffuse les données au client au fur et à mesure de leur production. Cependant, le client ne peut pas contrôler le rythme de livraison. Il n'existe pas d'équivalent `FetchSize` pour le point de terminaison HTTPS. Pour contrôler la consommation de mémoire côté client avec de grands ensembles de résultats, utilisez une connexion au pilote Bolt configurée. `FetchSize`

Configuration FetchSize et traitement des résultats par lots

Définissez `FetchSize` la session ou la configuration du pilote pour contrôler le nombre d'enregistrements que le pilote demande à Neptune par message PULL. Notez que cela `FetchSize` contrôle le flux pilote-serveur, et non le traitement par lots au niveau de l'application. Pour traiter les résultats par lots au niveau de l'application (par exemple, pour effectuer un travail intermédiaire entre des groupes d'enregistrements), utilisez une méthode d'assistance qui accumule les enregistrements provenant de l'itérateur paresseux du pilote.

Les exemples suivants montrent à la fois la configuration `FetchSize` sur la session et l'utilisation d'une méthode d'assistance pour regrouper les enregistrements en lots au niveau de l'application à des fins de traitement.

Example Java

```
static void processBatches(Result result, int size,
        Consumer<List<Record>> handler) {
```

```
    var batch = new ArrayList<Record>();
    while (result.hasNext()) {
        batch.add(result.next());
        if (batch.size() >= size) {
            handler.accept(batch);
            batch = new ArrayList<>();
        }
    }
    if (!batch.isEmpty()) handler.accept(batch);
}

// Usage:
try (Session session = driver.session(SessionConfig.builder()
        .withFetchSize(50)
        .withDefaultAccessMode(AccessMode.READ)
        .build())) {

    Result result = session.run("MATCH (m:movie) RETURN m.title AS title");
    processBatches(result, 10, batch -> {
        for (var record : batch) {
            System.out.println(record.get("title").asString());
        }

        // Do other intermediary work here between batch calls
    });
    result.consume();
}
```

Example Python

```
async def batched(result, size):
    batch = []
    async for record in result:
        batch.append(record)
        if len(batch) >= size:
            yield batch
            batch = []
    if batch:
        yield batch

# Usage:
async with driver.session(fetch_size=50) as session:
    result = await session.run("MATCH (m:movie) RETURN m.title AS title")
```

```
    async for batch in batched(result, 10):
        for record in batch:
            print(record["title"])

        # Do other intermediary work here between batch calls
```

Example Go

```
func processBatches(ctx context.Context, result neo4j.ResultWithContext,
        size int, handler func([]*neo4j.Record)) error {
    var batch []*neo4j.Record
    for result.Next(ctx) {
        batch = append(batch, result.Record())
        if len(batch) >= size {
            handler(batch)
            batch = nil
        }
    }
    if len(batch) > 0 {
        handler(batch)
    }
    return result.Err()
}

// Usage:
session := driver.NewSession(ctx, neo4j.SessionConfig{
    AccessMode: neo4j.AccessModeRead,
    FetchSize:  50,
})
defer session.Close(ctx)

result, err := session.Run(ctx,
    "MATCH (m:movie) RETURN m.title AS title", nil)
if err != nil {
    log.Fatal(err)
}

if err := processBatches(ctx, result, 10, func(batch []*neo4j.Record) {
    for _, record := range batch {
        title, _ := record.Get("title")
        fmt.Println(title)
    }
```

```
    // Do other intermediary work here between batch calls
}); err != nil {
    log.Fatal(err)
}
```

Example.NET

```
static async IAsyncEnumerable<List<IRecord>> Batched(
        IResultCursor result, int size)
{
    var batch = new List<IRecord>();
    while (await result.FetchAsync())
    {
        batch.Add(result.Current);
        if (batch.Count >= size)
        {
            yield return batch;
            batch = new List<IRecord>();
        }
    }
    if (batch.Count > 0) yield return batch;
}

// Usage:
await using var session = driver.AsyncSession(o => o
    .WithFetchSize(50)
    .WithDefaultAccessMode(AccessMode.Read));

var result = await session.RunAsync("MATCH (m:movie) RETURN m.title AS title");
await foreach (var batch in Batched(result, 10))
{
    foreach (var record in batch)
    {
        Console.WriteLine(record["title"].As<string>());
    }

    // Do other intermediary work here between batch calls
}
```

Example Node.js

```
// Uses for-await-of on the Result's async iterator (Neo4j driver v5+).
// This is pull-based: the driver pauses fetching while the loop body
```

```
// executes, providing natural backpressure. This is preferred over
// result.subscribe(), which is push-based and continues delivering
// records during async processing.
async function* batched(result, size) {
    let batch = [];
    for await (const record of result) {
        batch.push(record);
        if (batch.length >= size) {
            yield batch;
            batch = [];
        }
    }
    if (batch.length > 0) yield batch;
}

// Usage:
const session = driver.session({
    defaultAccessMode: neo4j.session.READ,
    fetchSize: 50
});

try {
    const result = session.run('MATCH (m:movie) RETURN m.title AS title');
    for await (const batch of batched(result, 10)) {
        for (const record of batch) {
            console.log(record.get('title'));
        }

        // Do other intermediary work here between batch calls
    }
} finally {
    await session.close();
}
```

Consommation rapide ou consommation incrémentielle

Les méthodes suivantes bloquent jusqu'à ce que la totalité du jeu de résultats soit collectée en mémoire, empêchant ainsi votre application d'agir sur les résultats lorsqu'ils arrivent :

- Java : `result.list()`
- Python :`result.data()`, `list(result)`
- Allez :`result.Collect(ctx)`, `neo4j.EagerResultTransformer`

- .NET : `result.ToListAsync()`
- Node.js: `const { records } = await result`

Pour traiter les résultats de manière incrémentielle, utilisez les modèles d'itération paresseuse et d'assistance par lots illustrés dans les exemples ci-dessus.

Conseils de requête OpenCypher

> **Important**
>
> L'indice de requête OpenCypher n'est disponible qu'à partir des versions 1.3.2.0 et ultérieures du moteur.

Dans Amazon Neptune, vous pouvez utiliser la `USING` clause pour spécifier des indices de requête pour les requêtes OpenCypher. Ces conseils vous permettent de contrôler les stratégies d'optimisation et d'évaluation.

La syntaxe des indices de requête est la suivante :

```
USING {scope}:{hint} {value}
```

1. `{scope}`définit la portée dans laquelle l'indice s'applique à : `Query` ou`Clause`.

   Une valeur de portée égale à `Query` signifie que l'indice de requête s'applique à l'ensemble de la requête (au niveau de la requête).

   Une valeur de portée égale à `Clause` signifie que l'indice de requête s'applique à la clause qu'il précède (au niveau de la clause).
2. `{hint}`est le nom de l'indice de requête appliqué.
3. `{value}`est l'argument en faveur du`{hint}`.

Les valeurs peuvent ne pas faire la distinction majuscules/majuscules.

Par exemple, pour activer le cache du plan de requête pour une requête :

```
Using QUERY:PLANCACHE "enabled"
MATCH (a:Person {firstName: "Erin", lastName: $lastName})
```

```
RETURN a
```

> **Note**
>
> Actuellement, la requête Query Scope indique que PLANCACHE, TIMEOUTMILLISECONDS et assume sont pris en charge. ConsistentDataTypes Les conseils de requête pris en charge sont répertoriés ci-dessous.

Rubriques

- Indication du cache du plan de requête OpenCypher
- AssumeConsistentDataTypes indice
- Indication du délai d'expiration de la requête OpenCypher

Indication du cache du plan de requête OpenCypher

Le comportement du cache du plan de requête peut être modifié par requête (paramétrée ou non) par un indice de requête au niveau de la requête. `QUERY:PLANCACHE` Il doit être utilisé avec la `USING` clause. L'indice de requête accepte `enabled` ou `disabled` en tant que valeur. Pour plus d'informations sur le cache du plan de requête, consultezCache du plan de requête dans Amazon Neptune.

AWS CLI

Forcer le plan à être mis en cache ou réutilisé :

```
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1"
```

Avec paramètres :

```
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "Using QUERY:PLANCACHE \"enabled\" RETURN \$arg" \
  --parameters '{"arg": 123}'
```

Forcer le plan à ne pas être mis en cache ni réutilisé :

```
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "Using QUERY:PLANCACHE \"disabled\" MATCH(n) RETURN n LIMIT 1"
```

Pour plus d'informations, voir execute-open-cypher-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

# Forcing plan to be cached or reused
response = client.execute_open_cypher_query(
    openCypherQuery='Using QUERY:PLANCACHE "enabled" MATCH(n) RETURN n LIMIT 1'
)

print(response['results'])
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

Forcer le plan à être mis en cache ou réutilisé :

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "query=Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

Forcer le plan à être mis en cache ou réutilisé :

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1"
```

Avec paramètres :

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" RETURN \$arg" \
  -d "parameters={\"arg\": 123}"
```

Forcer le plan à ne pas être mis en cache ni réutilisé :

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"disabled\" MATCH(n) RETURN n LIMIT 1"
```

AssumeConsistentDataTypes indice

OpenCypher suit un paradigme dans lequel les correspondances de types de données numériques (par exemple, int, octet, short, long, etc.) sont effectuées selon une sémantique de promotion de type. Par exemple, lorsque vous recherchez toutes les propriétés avec une valeur d'entrée 10 avec un type court, sous sémantique de promotion de type, cela correspond également aux propriétés dont la valeur longue est 10. Dans certains cas, le changement de type peut entraîner une surcharge et entraîner des plans de requêtes moins efficaces qu'ils ne le seraient si aucun changement de type n'était effectué. En particulier dans les cas où les types de données sont utilisés de manière cohérente dans les données (par exemple, si l'âge de toutes les personnes est stocké sous forme de valeur longue), l'exécution de promotions de type entraîne une surcharge sans impact sur le résultat de la requête.

Pour permettre l'optimisation dans les cas où l'on sait que les valeurs des données de propriétés numériques stockées dans la base de données sont de type cohérent, un indice de requête appelé `assumeConsistentDataTypes` (avec valeur`true/false`, la valeur par défaut étant`false`) peut être utilisé. Lorsque cet indice de requête est fourni avec une valeur égale à`true`, le moteur suppose que les seules valeurs de propriété sont toujours longues ou doubles et ignore la sémantique de promotion du type. Les valeurs numériques spécifiées dans la requête sont considérées comme des

valeurs longues (pour les valeurs non flottantes) ou comme des valeurs doubles (pour les valeurs à virgule flottante).

Si les données utilisent systématiquement un seul type de données (par exemple, tous les âges sont stockés sous forme`long`), l'utilisation de l'`assumeConsistentDataTypes`indice peut optimiser la requête en évitant les vérifications d'égalité inutiles pour les différents types numériques. Toutefois, si les données présentent des types de données incohérents pour la même propriété, l'utilisation de l'indice peut entraîner l'absence de certains résultats, car la requête ne correspondra qu'au seul type de données supposé par l'indice.

```
# Database loaded with following openCypher CSV's

# File 1
:ID,age:Int
n1,20
n2,25

# File 2
:ID,age:Long
n3,25


# Example (no hint)
MATCH (n:Person)
WHERE n.age >= 25
RETURN n

# Result
n2
n3

Returns all person whose age is >= 25 and the values >= 25 can be with any of these
 datatypes
i.e. byte, short, int, long, double or float

-------------------------------------------------------------------------------

# Example (with hint present)
USING QUERY:assumeConsistentDataTypes "true"
MATCH (n:Person)
WHERE n.age >= 25
RETURN n
```

```
# Result
n3

Returns only "n3" and not "n2". The reason is that even though the numerical value
matches (25), the datatype is "int" and is considered a non-match.
```

La différence peut également être validée via l'explication.

Sans l'explication :

```
# Query
MATCH (n)
WHERE n.age = 20
RETURN n

# Explain Snippet
################################################################################################
# ID # Out #1 # Out #2 # Name                    # Arguments

         # Mode # Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFEPipelineScan (DFX)  # pattern=Node(?n) with property 'age'
 as ?n_age2 and label 'ALL'
          # -    # 0        # 1         # 0.00  # 0.10      #
#    #        #        #                       # inlineFilters=[(?n_age2 IN
 ["20"^^xsd:byte, "20"^^xsd:int, "20"^^xsd:long, "20"^^xsd:short, "20.0"^^xsd:double,
 "20.0"^^xsd:float])] #      #          #           #       #           #
#    #        #        #                       # patternEstimate=1

         #      #          #           #       #           #
################################################################################################

# The inFilters field contains all numeric types
```

Avec l'indice :

```
# Query
MATCH (n)
WHERE n.age = 20
RETURN n

# Explain Snippet
################################################################################################
```

```
# ID # Out #1 # Out #2 # Name                    # Arguments
                           # Mode # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFEPipelineScan (DFX)  # pattern=Node(?n) with property 'age'
 as ?n_age2 and label 'ALL' # -    # 0        # 1         # 0.00  # 0.07      #
#    #        #        #                        # inlineFilters=[(?n_age2 IN
 ["20"^^xsd:long])]                  #      #          #           #       #
 #
#    #        #        #                        # patternEstimate=1
                           #      #          #           #       #           #
#########################################################################################

# The inFilters field only contains long datatype
```

Indication du délai d'expiration de la requête OpenCypher

Le comportement du délai d'expiration des requêtes peut être configuré par requête à l'aide d'un indice de requête au niveau de la requête. `QUERY:TIMEOUTMILLISECONDS` Il doit être utilisé avec la `USING` clause. L'indice de requête accepte une valeur longue non négative comme valeur.

AWS CLI

```
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "USING QUERY:TIMEOUTMILLISECONDS 100 MATCH(n) RETURN n LIMIT
 1"
```

Pour plus d'informations, voir execute-open-cypher-query dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_query(
    openCypherQuery='USING QUERY:TIMEOUTMILLISECONDS 100 MATCH(n) RETURN n LIMIT 1'
```

```
)

print(response['results'])
```

Pour des exemples de AWS SDK dans d'autres langues, voirAWS SDK.

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "query=USING QUERY:TIMEOUTMILLISECONDS 100 MATCH(n) RETURN n LIMIT 1"
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=USING QUERY:TIMEOUTMILLISECONDS 100 MATCH(n) RETURN n LIMIT 1"
```

Le comportement de temporisation des requêtes prendra en compte le minimum du délai d'expiration au niveau du cluster et du délai d'expiration au niveau de la requête. Consultez les exemples ci-dessous pour comprendre le comportement d'expiration des requêtes. Pour plus d'informations sur le délai d'expiration des requêtes au niveau du cluster, consultez neptune_query_timeout.

```
# Suppose `neptune_query_timeout` is 10000 ms and query-level timeout is set to 100 ms
# It will consider 100 ms as the final timeout

curl https://your-neptune-endpoint:port/openCypher \
  -d "query=USING QUERY:TIMEOUTMILLISECONDS 100 MATCH(n) RETURN n LIMIT 1"

# Suppose `neptune_query_timeout` is 100 ms and query-level timeout is set to 10000 ms
# It will still consider 100 ms as the final timeout

curl https://your-neptune-endpoint:port/openCypher \
```

```
  -d "query=USING QUERY:TIMEOUTMILLISECONDS 10000 MATCH(n) RETURN n LIMIT 1"
```

Si une requête dépasse le délai d'expiration, Neptune y met fin et renvoie une erreur de temporisation. L'opportunité de réessayer une requête expirée dépend de la nature de l'échec et de votre charge de travail. Pour de plus amples informations, consultez the section called “Exceptions et nouvelles tentatives”.

Limites Neptune openCypher

La version Amazon Neptune d'openCypher ne prend toujours pas en charge tout ce qui est spécifié dans la version 9 de Cypher Query Language Reference, comme indiqué dans Conformité aux spécifications OpenCypher. Les prochaines versions devraient remédier à un grand nombre de ces limitations.

Exceptions Neptune openCypher

Lorsque vous utilisez openCypher sur Amazon Neptune, diverses exceptions peuvent se produire. Vous trouverez ci-dessous les exceptions courantes que vous pouvez recevoir, soit du point de terminaison HTTPS, soit du pilote Bolt (toutes les exceptions du pilote Bolt sont signalées comme des exceptions d'état du serveur). Lorsque la solution consiste à réessayer la demande, reportez-vous à la section the section called “Exceptions et nouvelles tentatives” pour obtenir des conseils sur l'élaboration d'une stratégie de nouvelle tentative efficace.

Code HTTP	Message d'erreur	Nouvelle tentative possible ?	Solution	
400	(erreur de syntaxe, propagée directement depuis l'analyseur openCypher)	Non	Corrigez la syntaxe de la requête, puis réessayez.	
500	`Operation terminated (out of memory)`	Oui	Retravaillez la requête pour ajouter des critères	

Code HTTP	Message d'erreur	Nouvelle tentative possible ?	Solution	
			de filtrage supplémentaires afin de réduire la mémoire requise.	
500	Opération suspendue (délai dépassé)	Oui	Augmentez le délai d'expiration de la requête dans le groupe de paramètres du cluster de bases de données ou effectuez une nouvelle tentative.	
500	Opération suspendue (annulée par l'utilisateur)	Oui	Réitérez la demande.	
500	La réinitialisation de la base de données est en cours. Réessayez la requête une fois que le cluster sera disponible.	Oui	Réessayez une fois la réinitialisation terminée.	

Code HTTP	Message d'erreur	Nouvelle tentative possible ?	Solution
500	L'opération a échoué en raison d'opérations simultanées contradictoires (veuillez réessayer). Les transactions sont en cours d'annulation.	Oui	Réitérez la demande.
400	*(operation name)* operation/feature Exception non prise en charge	Non	L'opération spécifiée n'est pas prise en charge.
400	Tentative de mise à jour d'openCypher sur un réplica en lecture seule	Non	Remplacez le point de terminaison cible par le point de terminaison de l'enregistreur.
400	MalformedQueryException (Neptune n'affiche pas l'état interne de l'analyseur)	Non	Corrigez la syntaxe de la requête, puis réessayez.

Code HTTP	Message d'erreur	Nouvelle tentative possible ?	Solution	
400	Impossible de supprimer le nœud, car il possède toujours des relations. Pour supprimer ce nœud, vous devez d'abord supprimer ses relations.	Non	Au lieu de `MATCH (n) DELETE n`, utilisez `MATCH(n) DETACH DELETE(n)`	

Code HTTP	Message d'erreur	Nouvelle tentative possible ?	Solution	
400	Opération non valide : tentative de suppression de la dernière étiquette d'un nœud. Un nœud doit avoir au moins une étiquette.	Non	Neptune exige que tous les nœuds aient au moins une étiquette, et si des nœuds sont créés sans étiquette explicite, une étiquette par défaut `vertex` est attribuée . Modifiez la logique de and/or l'application de requête afin de ne pas supprimer le dernier libellé. L'étiquette singleton d'un nœud peut être mise à jour en définissant une nouvelle étiquette, puis en suppriman t l'ancienne étiquette.	

Code HTTP	Message d'erreur	Nouvelle tentative possible ?	Solution	
500	Nombre maximum de demandes dépassées, Configure dQueueCapacity = {} pour ConnID = {}	Oui	Actuellement, seules 8 192 demandes simultanées peuvent être traitées, quels que soient la pile et le protocole.	
500	Nombre maximal de connexions dépassé.	Oui	Seules 1 000 connexions Bolt simultanées par instance sont autorisées (pour le protocole HTTP, il n'y a pas de limite).	
400	[Nœud, relation ou chemin] attendu et obtention d'un littéral	Non	Vérifiez que vous transmettez le ou les arguments corrects, que la syntaxe de la requête est exacte, puis réessayez.	

Code HTTP	Message d'erreur	Nouvelle tentative possible ?	Solution	
400	La valeur de la propriété doit être un littéral simple. Ou : mappage attendu pour les propriétés de l'ensemble, mais mappage introuvable.	Non	Une clause SET n'accepte que les littéraux simples, et non les types composites.	
400	L'entité transmise pour la suppression est introuvable	Non	Vérifiez que l'entité que vous essayez de supprimer se trouve bien dans la base de données.	
400	L'utilisateur n'a pas accès à la base de données.	Non	Vérifiez la politique relative au rôle IAM utilisé.	
400	Aucun jeton n'est transmis dans le cadre de la demande	Non	Un jeton correctement signé doit être transmis dans le cadre de la demande de requête sur un cluster compatibl e IAM.	

Code HTTP	Message d'erreur	Nouvelle tentative possible ?	Solution	
400	Le message d'erreur est propagé.	Non	Contactez le AWS Support à l'aide de l'ID de demande.	
500	Opération suspendue (erreur interne)	Oui	Contactez le AWS Support à l'aide de l'ID de demande.	

Extensions OpenCypher dans Amazon Neptune

Amazon Neptune prend en charge la version 9 de référence de la spécification OpenCypher. Reportez-vous Conformité aux spécifications OpenCypher dans Amazon Neptune à Amazon Neptune pour plus de détails. En outre, Amazon Neptune prend en charge les fonctionnalités répertoriées ici. À moins que des versions spécifiques ne soient mentionnées, les fonctionnalités sont disponibles dans Neptune Database et Neptune Analytics.

Query-time Accès aux données S3

Disponible dans Neptune Database 1.4.7.0 et versions ultérieures.

Neptune prend en charge la `neptune.read()` fonction permettant de lire les données CSV ou Parquet d'Amazon S3 directement dans les requêtes OpenCypher. Contrairement au chargeur groupé qui importe les données avant de les interroger, il `neptune.read()` accède aux données Amazon S3 au moment de l'exécution de la requête.

Pour une documentation complète, voirneptune.read ().

La fonction Neptune-specific **`join ()`**

Disponible dans Neptune Database et Neptune Analytics.

Neptune implémente une fonction `join()` qui ne se trouve pas dans la spécification openCypher. Cela crée un littéral de chaîne à partir d'une liste de littéraux de chaîne et d'un délimiteur de chaîne. Deux arguments sont donc utilisés :

- Le premier argument est une liste de littéraux de chaîne.
- Le deuxième argument est le délimiteur de chaîne, qui peut avoir un, aucune ou plusieurs caractères.

Exemple :

```
join(["abc", "def", "ghi"], ", ")    // Returns "abc, def, ghi"
```

La fonction Neptune-specific **`removeKeyFromMap()`**

Disponible dans Neptune Database et Neptune Analytics.

Neptune implémente une fonction `removeKeyFromMap()` qui ne se trouve pas dans la spécification openCypher. Il supprime une clé spécifiée dans un mappage et renvoie le nouveau mappage généré.

La fonction accepte deux arguments :

- Le premier argument est le mappage à partir duquel la clé doit être supprimée.
- Le premier argument est la clé à supprimer du mappage.

Cette fonction `removeKeyFromMap()` est particulièrement utile dans les situations où vous souhaitez définir les valeurs d'un nœud ou d'une relation en déroulant une liste de mappage. Par exemple :

```
UNWIND [{`~id`: 'id1', name: 'john'}, {`~id`: 'id2', name: 'jim'}] as val
CREATE (n {`~id`: val.`~id`})
SET n = removeKeyFromMap(val, '~id')
```

Valeurs d'ID personnalisées pour les propriétés des nœuds et des relations

Disponible dans Neptune Database 1.2.0.2 et versions ultérieures, et dans Neptune Analytics.

À partir de la version 1.2.0.2 du moteur, Neptune a étendu la spécification openCypher afin que vous puissiez désormais spécifier les valeurs `id` des nœuds et des relations dans les clauses `CREATE`,

MERGE et MATCH. Cela vous permet d'attribuer des chaînes conviviales au lieu d'UUID générés par le système pour identifier les nœuds et les relations.

Dans Neptune Analytics, les valeurs d'identification personnalisées ne sont pas disponibles pour les bords.

> **Warning**
>
> Cette extension de la spécification openCypher est rétrocompatible, car ~id est désormais considéré comme un nom de propriété réservé. Si vous l'utilisez déjà ~id en tant que propriété dans vos données et requêtes, vous devez migrer la propriété existante vers une nouvelle clé de propriété et supprimer l'ancienne. Consultez Que faire si vous utilisez actuellement ~id comme propriété.

Voici un exemple montrant comment créer des nœuds et des relations dotés d'ID personnalisés :

```
CREATE (n {`~id`: 'fromNode', name: 'john'})
  -[:knows {`~id`: 'john-knows->jim', since: 2020}]
  ->(m {`~id`: 'toNode', name: 'jim'})
```

Si vous essayez de créer un ID personnalisé déjà utilisé, Neptune génère une erreur DuplicateDataException.

Voici un exemple d'utilisation d'ID personnalisé dans une clause MATCH :

```
MATCH (n {`~id`: 'id1'})
RETURN n
```

Voici un exemple d'utilisation d'ID personnalisés dans une clause MERGE :

```
MATCH (n {name: 'john'}), (m {name: 'jim'})
MERGE (n)-[r {`~id`: 'john->jim'}]->(m)
RETURN r
```

Que faire si vous utilisez actuellement **~id** comme propriété

Avec la version 1.2.0.2 du moteur, la clé ~id dans les clauses openCypher est désormais traitée comme id plutôt que comme propriété. Dès lors, si vous avez une propriété nommée ~id, il devient impossible d'y accéder.

Si vous utilisez une propriété `~id`, avant de passer à la version de moteur `1.2.0.2` ou à une version ultérieure, vous devez migrer la propriété `~id` existante vers une nouvelle clé de propriété, puis supprimer la propriété `~id`. Par exemple, la requête ci-dessous :

- Crée une propriété nommée 'newId' pour tous les nœuds,
- copie la valeur de la propriété '~id' dans la propriété 'newId'
- et supprime la propriété '~id' des données

```
MATCH (n)
WHERE exists(n.`~id`)
SET n.newId = n.`~id`
REMOVE n.`~id`
```

La même approche doit être adoptée pour toutes les relations dans les données qui ont une propriété `~id`.

Vous devrez également modifier toutes les requêtes que vous utilisez qui font référence à une propriété `~id`. Par exemple, cette requête :

```
MATCH (n)
WHERE n.`~id` = 'some-value'
RETURN n
```

... serait remplacée par ce qui suit :

```
MATCH (n)
WHERE n.newId = 'some-value'
RETURN n
```

Support des sous-requêtes CALL dans Neptune

Disponible dans Neptune Database 1.4.1.0 et versions ultérieures, et dans Neptune Analytics.

Amazon Neptune prend en charge `CALL` les sous-requêtes. Une `CALL` sous-requête est une partie de la requête principale qui s'exécute dans une portée isolée pour chaque entrée de la `CALL` sous-requête.

Supposons, par exemple, qu'un graphique contienne des données sur les personnes, leurs amis et les villes dans lesquelles elles vivaient. Nous pouvons retrouver les deux plus grandes villes où vivait chaque ami d'une personne en utilisant une CALL sous-requête :

```
MATCH (person:Person)-[:knows]->(friend)
CALL {
  WITH friend
  MATCH (friend)-[:lived_in]->(city)
  RETURN city
  ORDER BY city.population DESC
  LIMIT 2
}
RETURN person, friend, city
```

Dans cet exemple, la partie requête interne `CALL { ... }` est exécutée pour chaque `friend` élément correspondant à la clause MATCH précédente. Lorsque la requête interne est exécutée, les `LIMIT` clauses `ORDER` and sont locales aux villes où vivait un ami en particulier. Nous obtenons donc (au maximum) deux villes par ami.

Toutes les clauses de requête sont disponibles dans les `CALL` sous-requêtes. Cela inclut également les `CALL` sous-requêtes imbriquées. Certaines restrictions relatives à la première `WITH` clause et aux variables émises existent et sont expliquées ci-dessous.

Étendue des variables dans la sous-requête CALL

Les variables des clauses situées avant la `CALL` sous-requête et utilisées à l'intérieur de celle-ci doivent être importées par la `WITH` clause initiale. Contrairement aux `WITH` clauses ordinaires, elle ne peut contenir qu'une liste de variables, mais elle n'autorise pas l'aliasing et ne peut pas être utilisée avec `DISTINCTORDER BY`,`WHERE`,`SKIP`, ou`LIMIT`.

Variables renvoyées par la sous-requête CALL

Les variables émises par la `CALL` sous-requête sont spécifiées dans la `RETURN` clause finale. Notez que les variables émises ne peuvent pas se chevaucher avec les variables avant la `CALL` sous-requête.

Limitations

À l'heure actuelle, les mises à jour au sein d'une `CALL` sous-requête ne sont pas prises en charge.

Fonctions Neptune OpenCypher

Disponible dans Neptune Database 1.4.1.0 et versions ultérieures, et dans Neptune Analytics.

texte IndexOf

`textIndexOf(text :: STRING, lookup :: STRING, from = 0 :: INTEGER?, to = -1 :: INTEGER?) :: (INTEGER?)`

Renvoie l'indice de la première occurrence comprise entre le `text` décalage `from` (inclus) et le décalage `to` (exclusif). `lookup` Si la `to` valeur est -1, la plage continue jusqu'à la fin de`text`. L'indexation est basée sur zéro et est exprimée en valeurs scalaires Unicode (points de code non substituts).

```
RETURN textIndexOf('Amazon Neptune', 'e')
{
  "results": [{
      "textIndexOf('Amazon Neptune', 'e')": 8
    }]
}
```

coll. ToSet

`collToSet(values :: LIST OF ANY?) :: (LIST? OF ANY?)`

Renvoie une nouvelle liste contenant uniquement les éléments uniques de la liste d'origine. L'ordre de la liste d'origine est conservé (par exemple, les `[1, 6, 5, 1, 5]` retours`[1, 6, 5]`).

```
RETURN collToSet([1, 6, 5, 1, 1, 5])
{
  "results": [{
      "collToSet([1, 6, 5, 1, 1, 5])": [1, 6, 5]
    }]
}
```

Soustract de Coll

`collSubtract(first :: LIST OF ANY?, second :: LIST OF ANY?) :: (LIST? OF ANY?)`

Renvoie une nouvelle liste contenant tous les éléments uniques dont les éléments sont `first` exclus`second`.

```
RETURN collSubtract([2, 5, 1, 0], [1, 5])
{
  "results": [{
      "collSubtract([2, 5, 1, 0], [1, 5])": [0, 2]
    }]
}
```

Intersection de Coll

`collIntersection(first :: LIST? OF ANY?, second :: LIST? OF ANY?) :: (LIST? OF ANY?)`

Renvoie une nouvelle liste contenant tous les éléments uniques de l'intersection de `first` et`second`.

```
RETURN collIntersection([2, 5, 1, 0], [1, 5])
{
  "results": [{
      "collIntersection([2, 5, 1, 0], [1, 5])": [1, 5]
    }]
}
```

Fonctions de tri

Les sections suivantes définissent les fonctions permettant de trier les collections. Ces fonctions utilisent (dans certains cas facultatifs) des arguments de `config` carte, ou une liste de plusieurs cartes de ce type, qui définissent la clé de tri et and/or le sens du tri :

```
{ key: STRING, order: STRING }
```

`key`Voici une propriété de carte ou de nœud dont la valeur doit être utilisée pour le tri. `order`est « » ou `asc` « `desc` » (sans distinction majuscules/minuscules) pour spécifier un tri croissant ou décroissant, respectivement. Par défaut, le tri sera effectué par ordre croissant.

Coll Sort

`collSort(coll :: LIST OF ANY, config :: MAP?) :: (LIST? OF ANY?)`

Renvoie une nouvelle liste triée contenant les éléments de la liste `coll` d'entrée.

```
RETURN collSort([5, 3, 1], {order: 'asc'})
```

```
{
  "results": [{
      "collSort([5, 3, 1])": [1, 3, 5]
    }]
}
```

coll. SortMaps

`collSortMaps(coll :: LIST OF MAP, config :: MAP) :: (LIST? OF ANY?)`

Renvoie une liste de cartes triées en fonction de la valeur de la `key` propriété spécifiée.

```
RETURN collSortMaps([{name: 'Alice', age: 25}, {name: 'Bob', age: 35}, {name:
 'Charlie', age: 18}], {key: 'age', order: 'desc'})
{
  "results": [{
      "x": [{
          "age": 35,
          "name": "Bob"
        }, {
          "age": 25,
          "name": "Alice"
        }, {
          "age": 18,
          "name": "Charlie"
        }]
    }]
}
```

coll. SortMulti

```
collSortMulti(coll :: LIST OF MAP?,
configs = [] :: LIST OF MAP,
limit = -1 :: INTEGER?,
skip = 0 :: INTEGER?) :: (LIST? OF ANY?)
```

Renvoie une liste de cartes triées en fonction de la valeur des `key` propriétés spécifiées, en appliquant éventuellement une limite et un saut.

```
RETURN collSortMulti([{name: 'Alice', age: 25}, {name: 'Bob', age: 35}, {name:
 'Charlie', age: 18}], [{key: 'age', order: 'desc'}, {key:'name'}]) as x
{
  "results": [{
```

```
      "x": [{
          "age": 35,
          "name": "Bob"
        }, {
          "age": 25,
          "name": "Alice"
        }, {
          "age": 18,
          "name": "Charlie"
        }]
    }]
}
```

coll. SortNodes

`collSortNodes(coll :: LIST OF NODE, config :: MAP) :: (LIST? OF NODE?)`

Renvoie une version triée de la liste `coll` d'entrée, triant les éléments du nœud en fonction des valeurs de leurs `key` propriétés respectives.

```
create (n:person {name: 'Alice', age: 23}), (m:person {name: 'Eve', age: 21}),
 (o:person {name:'Bob', age:25})
{"results":[]}

match (n:person) with collect(n) as people return collSortNodes(people, {key: 'name',
 order: 'desc'})
{
  "results": [{
      "collSortNodes(people, 'name')": [{
          "~id": "e599240a-8c23-4337-8aa8-f603c8fb5488",
          "~entityType": "node",
          "~labels": ["person"],
          "~properties": {
            "age": 21,
            "name": "Eve"
          }
        }, {
          "~id": "8a6ef785-59e3-4a0b-a0ff-389655a9c4e6",
          "~entityType": "node",
          "~labels": ["person"],
          "~properties": {
            "age": 25,
            "name": "Bob"
```

```
          }
        }, {
          "~id": "466bc826-f47f-452c-8a27-6b7bdf7ae9b4",
          "~entityType": "node",
          "~labels": ["person"],
          "~properties": {
            "age": 23,
            "name": "Alice"
          }
        }]
    }]
}

match (n:person) with collect(n) as people return collSortNodes(people, {key: 'age'})
{
  "results": [{
      "collSortNodes(people, '^age')": [{
          "~id": "e599240a-8c23-4337-8aa8-f603c8fb5488",
          "~entityType": "node",
          "~labels": ["person"],
          "~properties": {
            "age": 21,
            "name": "Eve"
          }
        }, {
          "~id": "466bc826-f47f-452c-8a27-6b7bdf7ae9b4",
          "~entityType": "node",
          "~labels": ["person"],
          "~properties": {
            "age": 23,
            "name": "Alice"
          }
        }, {
          "~id": "8a6ef785-59e3-4a0b-a0ff-389655a9c4e6",
          "~entityType": "node",
          "~labels": ["person"],
          "~properties": {
            "age": 25,
            "name": "Bob"
          }
        }]
    }]
}
```

Fonctions temporelles

Les fonctions temporelles sont disponibles à partir de Neptune 1.4.5.0.

day

`day(temporal :: (datetime | date)) :: (LONG)`

Renvoie le `day` mois à partir d'une `date` valeur `datetime` ou. Pour `datetime` : les valeurs sont normalisées en UTC en fonction des entrées avant d'extraire le jour. Pour `date` : le jour est extrait en fonction du fuseau horaire.

L'`datetime`entrée est disponible à la fois dans Neptune Database et Neptune Analytics :

```
RETURN day(datetime('2021-06-03T01:48:14Z'))
{
  "results": [{
      "day(datetime('2021-06-03T01:48:14Z'))": 3
    }]
}
```

Ici, le `datetime` temps est normalisé en UTC, donc + 08:00 revient au 2 juin.

```
RETURN day(datetime('2021-06-03T00:00:00+08:00'))
{
  "results": [{
      "day(datetime('2021-06-03T00:00:00+08:00'))": 2
    }]
}
```

L'`date`entrée n'est disponible que dans Neptune Analytics :

```
RETURN day(date('2021-06-03Z'))
{
  "results": [{
      "day(date('2021-06-03Z'))": 3
    }]
}
```

Le fuseau horaire `date` préserve, en gardant le 3 juin.

```
RETURN day(date('2021-06-03+08:00'))
```

```
{
  "results": [{
      "day(date('2021-06-03+08:00'))": 3
    }]
}
```

month

`month(temporal :: (datetime | date)) :: (LONG)`

Renvoie le mois à partir d'une `date` valeur `datetime` ou (1-12). Pour `datetime` : les valeurs sont normalisées en UTC en fonction des entrées avant d'extraire le mois. Pour `date` : le mois est extrait en fonction du fuseau horaire.

L'`datetime`entrée est disponible à la fois dans Neptune Database et Neptune Analytics :

```
RETURN month(datetime('2021-06-03T01:48:14Z'))
{
  "results": [{
      "month(datetime('2021-06-03T01:48:14Z'))": 6
    }]
}
```

Ici, le `datetime` temps est normalisé en UTC, donc + 08:00 revient au 31 mai.

```
RETURN month(datetime('2021-06-01T00:00:00+08:00'))
{
  "results": [{
      "month(datetime('2021-06-01T00:00:00+08:00'))": 5
    }]
}
```

L'`date`entrée n'est disponible que dans Neptune Analytics :

```
RETURN month(date('2021-06-03Z'))
{
  "results": [{
      "month(date('2021-06-03Z'))": 6
    }]
}
```

Le fuseau horaire `date` préserve, en gardant le 1er juin.

```
RETURN month(date('2021-06-01+08:00'))
{
  "results": [{
      "month(date('2021-06-01+08:00'))": 6
    }]
}
```

year

`year(temporal :: (datetime | date)) :: (LONG)`

Renvoie l'année à partir d'une `date` valeur `datetime` ou. Pour `datetime` : les valeurs sont normalisées en UTC sur la base des données saisies avant d'extraire l'année. Pour `date` : l'année est extraite en fonction du fuseau horaire.

L'`datetime`entrée est disponible à la fois dans Neptune Database et Neptune Analytics :

```
RETURN year(datetime('2021-06-03T01:48:14Z'))
{
  "results": [{
      "year(datetime('2021-06-03T01:48:14Z'))": 2021
    }]
}
```

Ici, le `datetime` temps est normalisé en UTC, donc + 08:00 revient au 31 décembre 2020.

```
RETURN year(datetime('2021-01-01T00:00:00+08:00'))
{
  "results": [{
      "year(datetime('2021-01-01T00:00:00+08:00'))": 2020
    }]
}
```

L'`date`entrée n'est disponible que dans Neptune Analytics :

```
RETURN year(date('2021-06-03Z'))
{
  "results": [{
      "year(date('2021-06-03Z'))": 2021
```

```
    }]
}
```

Le fuseau horaire `date` préserve le mois de juin 2021.

```
RETURN year(date('2021-01-01+08:00'))
{
  "results": [{
      "year(date('2021-01-01+08:00'))": 2021
    }]
}
```

Fonctions Neptune OpenCypher

Disponible dans Neptune Database 1.4.6.0 et versions ultérieures, et dans Neptune Analytics.

réduire ()

Reduce traite séquentiellement chaque élément de la liste en le combinant avec un total cumulé ou un « accumulateur ». En commençant par une valeur initiale, il met à jour l'accumulateur après chaque opération et utilise cette valeur mise à jour lors de l'itération suivante.

`for i in (0, ..., n) acc = acc X list[I], where X denotes any binary operator`

Une fois que tous les éléments ont été traités, il renvoie le résultat final cumulé.

Une structure de réduction () typique serait - `reduce(accumulator = initial , variable IN list | expression)`

Spécifications du type :

`- initial: starting value for the accumulator :: (Long | FLOAT | STRING | LIST? OF (STRING, LONG, FLOAT)) - list: the input list :: LIST OF T where T matches initial type - variable :: represents each element in the input list - expression :: Only supports '+' and '*' operator - return :: Same type as initial`

Restrictions :

Actuellement, l'`reduce( )`expression ne prend en charge que :

- Multiplication numérique

- Addition numérique
- Concaténation de chaînes
- Concaténation de listes

Ils sont représentés par l'*opérateur + or. L'expression doit être une expression binaire comme indiqué ci-dessous - `expression pattern: accumulator + any variable or accumulator * any variable`

Gestion des débordements :

Neptune détecte un dépassement numérique lors de l'`reduce( )`évaluation et réagit différemment en fonction du type de données :

```
LONG (signed#64-bit)
--------------------
• Valid range: -9#223#372#036#854#775#808#…#9#223#372#036#854#775#807
• If any intermediate or final value falls outside this range,
  Neptune aborts the query with long overflow error message.

FLOAT (IEEE-754 double)
-----------------------
• Largest finite value ≈#1.79#×#10^308
• Larger results overflow to INF
  Once `INF` is produced, it propagates through the remainder
  of the reduction.
```

Exemples :

Consultez les exemples suivants pour la fonction reduce ().

```


1. Long Addition:
RETURN reduce(sum = 0, n IN [1, 2, 3] | sum + n)
{
  "results": [{
      "reduce(sum = 0, n IN [1, 2, 3] | sum + n)": 6
    }]
}

2. String Concatenation:
RETURN reduce(str = "", x IN ["A", "B", "C"] | str + x)
```

```
{
  "results": [{
      "reduce(str = "", x IN ["A", "B", "C"] | str + x)": "ABC"
    }]
}

3. List Combination:
RETURN reduce(lst = [], x IN [1, 2, 3] | lst + x)
{
  "results": [{
      "reduce(lst = [], x IN [1, 2, 3] | lst + x)": [1, 2, 3]
    }]
}

4. Float Addition:
RETURN reduce(total = 0.0, x IN [1.5, 2.5, 3.5] | total + x)
{
  "results": [{
      "reduce(total = 0.0, x IN [1.5, 2.5, 3.5] | total + x)": 7.5
    }]
}

5. Long Multiplication:
RETURN reduce(product = 1, n IN [1, 2, 3] | product * n)
{
  "results": [{
      "reduce(product = 0, n IN [1, 2, 3] | product * n)": 6
    }]
}

6. Float Multiplication:
RETURN reduce(product = 1.0, n IN [1.5, 2.5, 3.5] | product * n)
{
  "results": [{
      "reduce(product = 1.0, n IN [1.5, 2.5, 3.5] | product * n)": 13.125
    }]
}

7. Long Overflow (Exception):
RETURN reduce(s = 9223372036854775807, x IN [2, 3] | s * x) AS result
{
"results": [{
    "reduce(s = 9223372036854775807, x IN [2, 3] | s * x) AS result": long overflow
    }]
```

```
}

8. Float Overflow:
RETURN reduce(s = 9.0e307, x IN [8.0e307, 1.0e307] | s + x) AS result
{
"results": [{
    "reduce(s = 9.0e307, x IN [8.0e307, 1.0e307] | s + x) AS result": INF
    }]
}
```

neptune.read ()

Neptune prend en charge une `CALL` procédure `neptune.read` permettant de lire les données d'Amazon S3, puis d'exécuter une requête OpenCypher (lecture, insertion, mise à jour) à l'aide de ces données. La procédure génère chaque ligne du fichier en tant que ligne de variable de résultat déclarée. Il utilise les informations d'identification IAM de l'appelant pour accéder aux données dans Amazon S3. Consultez Gérer les autorisations pour neptune.read () pour configurer les autorisations. La AWS région du compartiment Amazon S3 doit se trouver dans la même région que celle où se trouve l'instance. Actuellement, les lectures entre régions ne sont pas prises en charge.

Syntaxe

```
CALL neptune.read(
  {
    source: "string",
    format: "parquet/csv",
    concurrency: 10
  }
)
YIELD row
...
```

Inputs

- source (obligatoire) - URI Amazon S3 vers un seul objet. Le préfixe Amazon S3 désignant plusieurs objets n'est pas pris en charge.
- format (obligatoire) - `parquet` et `csv` sont pris en charge.
  - Vous trouverez plus de détails sur le format Parquet pris en charge dansTypes de colonnes de parquet pris en charge.

  - Pour plus d'informations sur le format csv pris en charge, consultezFormat de chargement de données Gremlin.
- simultanéité (facultatif) - Type : entier égal ou supérieur à 0. Valeur par défaut : 0. Spécifie le nombre de threads à utiliser pour lire le fichier. Si la valeur est 0, le nombre maximum de threads autorisés par la ressource sera utilisé. Pour le parquet, il est recommandé de définir un certain nombre de groupes de lignes.

Sorties

Le fichier neptune.read renvoie :

- ligne - Type : carte
  - Chaque ligne du fichier, où les clés sont les colonnes et les valeurs sont les données présentes dans chaque colonne.
  - Vous pouvez accéder aux données de chaque colonne sous la forme d'un accès aux propriétés (`row.col`).

Meilleures pratiques pour neptune.read ()

Les opérations de lecture de Neptune S3 peuvent être gourmandes en mémoire. Veuillez utiliser des types d'instance adaptés aux charges de travail de production, comme indiqué dans la section Choix des types d'instance pour Amazon Neptune.

L'utilisation de la mémoire et les performances des `neptune.read()` requêtes sont affectées par divers facteurs tels que la taille du fichier, le nombre de colonnes, le nombre de lignes et le format du fichier. Selon la structure, les petits fichiers (par exemple, les fichiers CSV de 100 Mo ou moins, les fichiers Parquet de 20 Mo ou moins) peuvent fonctionner de manière fiable sur la plupart des types d'instances adaptés à la production, tandis que les fichiers plus volumineux peuvent nécessiter une mémoire importante que les types d'instances plus petits ne peuvent pas fournir.

Lorsque vous testez cette fonctionnalité, il est recommandé de commencer par de petits fichiers et de procéder à une mise à l'échelle progressive afin de garantir que votre charge de travail de lecture puisse être adaptée à la taille de votre instance. Si vous remarquez que des `neptune.read()` demandes entraînent des exceptions liées à un manque de mémoire ou des redémarrages d'instances, envisagez de diviser vos fichiers en plus petits morceaux, de réduire la complexité des fichiers ou de passer à des types d'instances plus importants.

Exemples de requêtes utilisant du parquet

L'exemple de requête suivant renvoie le nombre de lignes d'un fichier Parquet donné :

```
CALL neptune.read(
  {
    source: "<s3 path>",
    format: "parquet"
  }
)
YIELD row
RETURN count(row)
```

Vous pouvez exécuter l'exemple de requête à l'aide de l'`execute-open-cypher-query`opération décrite dans le en AWS CLI exécutant le code suivant :

```
aws neptunedata execute-open-cypher-query \
--open-cypher-query "CALL neptune.read({source: '<s3 path>', format: 'parquet'}) YIELD
 row RETURN count(row)" \
--endpoint-url https://my-cluster-name.cluster-abcdefgh1234.us-
east-1.neptune.amazonaws.com:8182
```

Une requête peut être flexible quant à l'utilisation des lignes lues à partir d'un fichier Parquet. Par exemple, la requête suivante crée un nœud dont le champ est défini sur la base des données trouvées dans le fichier Parquet :

```
CALL neptune.read(
  {
    source: "<s3 path>",
    format: "parquet"
  }
)
YIELD row
CREATE (n {someField: row.someCol})
RETURN n
```

> **Warning**
>
> Il n'est pas considéré comme une bonne pratique d'utiliser une clause volumineuse produisant des résultats comme `MATCH(n)` avant une `CALL` clause. Cela entraînerait une requête de longue durée, en raison d'un produit croisé entre les solutions entrantes issues

des clauses précédentes et les lignes lues par neptune.read. Il est recommandé de démarrer la requête avec `CALL neptune.read`.

Types de colonnes de parquet pris en charge

Types de données sur le parquet :

- NULL
- BOOLEAN
- FLOAT
- DOUBLE
- CHAÎNE
- ENTIER SIGNÉ : UINT8, UINT16, UINT32, UINT64
- MAP : ne prend en charge qu'un seul niveau. Ne prend pas en charge le mode imbriqué.
- LISTE : ne prend en charge qu'un seul niveau. Ne prend pas en charge le mode imbriqué.

Neptune-specific types de données :

Contrairement aux en-têtes de colonne de propriétés au format CSV, les en-têtes de colonne de propriétés du format Parquet doivent uniquement contenir les noms des propriétés, il n'est donc pas nécessaire d'avoir les noms des types ni la cardinalité.

Certains types de colonnes spéciaux au format Parquet nécessitent toutefois une annotation dans les métadonnées, notamment le type Any, le type Date, le type DateTime et le type Geometry. L'objet suivant est un exemple de l'annotation de métadonnées requise pour les fichiers contenant des colonnes de ces types spéciaux :

```
"metadata": {
    "anyTypeColumns": ["UserCol1"],
    "dateTypeColumns": ["UserCol2"],
    "dateTimeTypeColumns": ["UserCol3"],
    "geometryTypeColumns": ["UserCol4"]
}
```

Vous trouverez ci-dessous des informations détaillées sur la charge utile attendue associée à ces types :

- Un type de colonne Any est pris en charge dans les colonnes utilisateur. Un type quelconque est un « sucre syntaxique » pour tous les autres types que nous prenons en charge. C'est extrêmement utile si une colonne utilisateur contient plusieurs types. La charge utile d'une valeur de type Any est une liste de chaînes json comme suit :`{"value": "10", "type": "Int"}; {"value": "1.0", "type": "Float"}`, qui contient un champ de valeur et un champ de type dans chaque chaîne json individuelle. La valeur de cardinalité d'une colonne Any est définie, ce qui signifie que la colonne peut accepter plusieurs valeurs.
  - Neptune prend en charge les types suivants dans tous les types : Bool (ou Boolean), Byte, Short, Int, Long,,,,, Float UnsignedByte UnsignedShort, Double UnsignedInt, Date UnsignedLong, DateTime, String et Geometry.
  - Le type de vecteur n'est pris en charge dans aucun type.
  - Nested Tous les types ne sont pas pris en charge. Par exemple, `{"value": {"value": "10", "type": "Int"}, "type": "Any"}`.
- Les colonnes de type Date et Datetime sont prises en charge dans les colonnes utilisateur. La charge utile de ces colonnes doit être fournie sous forme de chaînes suivant le format XSD ou l'un des formats ci-dessous :
  - aaaa- MM-dd
  - aaaa- MM-ddTHH:mm
  - aaaa- MM-ddTHH:mm:ss
  - aaaa- MM-ddTHH:mm:ssZ
  - aaaa- MM-ddTHH:mm:ss.SSSZ
  - yyyy- MM-ddTHH:mm:ss [+|-] hhmm
  - yyyy- MM-ddTHH:mm:ss.SSS [+|-] hhmm
- Un type de colonne de géométrie est pris en charge dans les colonnes utilisateur. La charge utile de ces colonnes ne doit contenir que des primitives de géométrie de type Point, fournies sous forme de chaînes au format Well-known texte (WKT). Par exemple, POINT (30 10) serait une valeur de géométrie valide.

Exemple de sortie pour parquet

Étant donné un fichier Parquet comme celui-ci :

```
<s3 path>

Parquet Type:
```

```
    int8     int16       int32            int64              float      double
 string
+--------+---------+------------+--------------------+-----------+-----------
+----------+
|   Byte |   Short |       Int  |               Long |     Float |    Double |
 String   |
|--------+---------+------------+--------------------+-----------+-----------
+----------|
|   -128 |  -32768 | -2147483648 | -9223372036854775808 |    1.23456 |    1.23457 |
 first    |
|    127 |   32767 |  2147483647 |  9223372036854775807 |  nan       |  nan       |
 second   |
|      0 |       0 |           0 |                    0 | -inf       | -inf       |
 third    |
|      0 |       0 |           0 |                    0 |  inf       |  inf       |
 fourth   |
+--------+---------+------------+--------------------+-----------+-----------
+----------+
```

Voici un exemple de sortie renvoyée par neptune.read à l'aide de la requête suivante :

```
aws neptunedata execute-open-cypher-query \
--open-cypher-query "CALL neptune.read({source: '<s3 path>', format: 'parquet'}) YIELD
 row RETURN row" \
--endpoint-url https://my-cluster-name.cluster-abcdefgh1234.us-
east-1.neptune.amazonaws.com:8182
```


```
{
 "results": [{
 "row": {
 "Float": 1.23456,
 "Byte": -128,
 "Int": -2147483648,
 "Long": -9223372036854775808,
 "String": "first",
 "Short": -32768,
 "Double": 1.2345678899999999
 }
 }, {
 "row": {
 "Float": "NaN",
 "Byte": 127,
 "Int": 2147483647,
```

```
 "Long": 9223372036854775807,
 "String": "second",
 "Short": 32767,
 "Double": "NaN"
 }
 }, {
 "row": {
 "Float": "-INF",
 "Byte": 0,
 "Int": 0,
 "Long": 0,
 "String": "third",
 "Short": 0,
 "Double": "-INF"
 }
 }, {
 "row": {
 "Float": "INF",
 "Byte": 0,
 "Int": 0,
 "Long": 0,
 "String": "fourth",
 "Short": 0,
 "Double": "INF"
 }
 }]
}
```

Il n'existe actuellement aucun moyen de définir une étiquette de nœud ou de bord pour un champ de données provenant d'un fichier Parquet. Il est recommandé de partitionner les requêtes en plusieurs requêtes, une pour chacune label/Type.

```
CALL neptune.read({source: '<s3 path>', format: 'parquet'})
 YIELD row
WHERE row.`~label` = 'airport'
CREATE (n:airport)

CALL neptune.read({source: '<s3 path>', format: 'parquet'})
YIELD row
WHERE row.`~label` = 'country'
CREATE (n:country)
```

Exemples de requêtes au format CSV

Dans cet exemple, la requête renvoie le nombre de lignes d'un fichier CSV donné :

```
CALL neptune.read(
  {
    source: "<s3 path>",
    format: "csv"
  }
)
YIELD row
RETURN count(row)
```

Vous pouvez exécuter l'exemple de requête à l'aide de l'opération execute-open-cypher-query dans le en exécutant le AWS CLI code suivant :

```
aws neptunedata execute-open-cypher-query \
--open-cypher-query "CALL neptune.read({source: '<s3 path>', format: 'csv'}) YIELD row
 RETURN count(row)" \
--endpoint-url https://my-cluster-name.cluster-abcdefgh1234.us-
east-1.neptune.amazonaws.com:8182
```

Une requête peut être flexible quant à l'utilisation des lignes lues à partir d'un fichier CSV. Par exemple, la requête suivante crée un nœud avec un champ défini sur les données d'un fichier CSV :

```
CALL neptune.read(
  {
    source: "<s3 path>",
    format: "csv"
  }
)
YIELD row
CREATE (n {someField: row.someCol})
RETURN n
```

> **Warning**
>
> Il n'est pas considéré comme une bonne pratique d'utiliser une clause de grande taille produisant des résultats telle que MATCH (n) avant une clause CALL. Cela entraînerait une requête de longue durée en raison d'un produit croisé entre les solutions entrantes issues des

clauses précédentes et les lignes lues par neptune.read. Il est recommandé de démarrer la requête avec CALL neptune.read.

En-têtes de colonnes de propriétés

Vous pouvez spécifier une colonne (`:`) pour une propriété à l'aide de la syntaxe suivante. Les noms de type ne sont pas sensibles à la casse. Si deux points apparaissent dans le nom d'une propriété, il faut l'éviter en le faisant précéder d'une barre oblique inverse :. `\:`

```
propertyname:type
```

Note

- Les espaces, les virgules, le retour en chariot et les caractères de nouvelle ligne ne sont pas autorisés dans les en-têtes de colonne. Les noms de propriétés ne peuvent donc pas inclure ces caractères.
- Vous pouvez spécifier une colonne pour un type de tableau en ajoutant `[]` au type :

  ```
  propertyname:type[]
  ```

- Les propriétés d'arc ne peuvent avoir qu'une seule valeur et provoquent une erreur si un type de tableau ou une seconde valeur est spécifié. L'exemple suivant montre l'en-tête de colonne d'une propriété nommée age de type Int :

  ```
  age:Int
  ```

Chaque ligne du fichier doit obligatoirement avoir un nombre entier dans cette position ou rester vide. Les tableaux de chaînes sont autorisés, mais les chaînes d'un tableau ne peuvent pas inclure le point-virgule (`;`) à moins qu'il ne soit évité par une barre oblique inverse (). `\;`

Types de colonnes CSV pris en charge

- BOOL (ou BOOLEAN) - Valeurs autorisées : true, false. Indique un champ booléen. Toute valeur autre que true sera traitée comme fausse.

- FLOAT - Plage : virgule flottante IEEE 754 32 bits, y compris Infinity, INF, -Infinity, -INF et NaN (pas un chiffre).
- DOUBLE : plage : virgule flottante IEEE 754 64 bits, y compris Infinity, INF, -Infinity, -INF et NaN (pas un chiffre).
- CHAÎNE -
  - Les guillemets sont facultatifs. Les virgules, les nouvelles lignes et les caractères renvoyant le chariot sont automatiquement ignorés s'ils sont inclus dans une chaîne entourée de guillemets doubles («). Exemple : « Bonjour tout le monde ».
  - Pour inclure des guillemets dans une chaîne entre guillemets, vous pouvez éviter les guillemets en utilisant deux guillemets d'affilée : « Hello «" World" "».
  - Les tableaux de chaînes sont autorisés, mais les chaînes d'un tableau ne peuvent pas inclure le point-virgule (;) à moins qu'il ne soit évité par une barre oblique inverse (\ ;).
  - Si vous souhaitez placer des chaînes d'un tableau entre guillemets, vous devez entourer la totalité du tableau par un ensemble de guillemets. Exemple : « Chaîne 1 ; Chaîne 2 ; Chaîne 3 ».
- DATE, DATETIME - Les valeurs de date/heure peuvent être fournies au format XSD ou dans l'un des formats suivants :
  - aaaa- MM-dd
  - aaaa- MM-ddTHH:mm
  - aaaa- MM-ddTHH:mm:ss
  - aaaa- MM-ddTHH:mm:ssZ
  - aaaa- MM-ddTHH:mm:ss.SSSZ
  - yyyy- MM-ddTHH:mm:ss [+|-] hhmm
  - yyyy- MM-ddTHH:mm:ss.SSS [+|-] hhmm
- ENTIER SIGNÉ -
  - Octet : -128 à 127
  - Court : -32768 à 32767
  - Int : -2^31 à 2^31-1
  - Longue : -2^63 à 2^63-1

Neptune-specific types de colonnes :

- Un type de colonne Any est pris en charge dans les colonnes utilisateur. Un type quelconque est un « sucre syntaxique » pour tous les autres types que nous prenons en charge. C'est

extrêmement utile si une colonne utilisateur contient plusieurs types. La charge utile d'une valeur de type Any est une liste de chaînes json comme suit :`{"value": "10", "type": "Int"}; {"value": "1.0", "type": "Float"}`, qui contient un champ de valeur et un champ de type dans chaque chaîne json individuelle. L'en-tête de colonne d'un type Any est PropertyName:Any. La valeur de cardinalité d'une colonne Any est définie, ce qui signifie que la colonne peut accepter plusieurs valeurs.

  - Neptune prend en charge les types suivants dans tous les types : Bool (ou Boolean), Byte, Short, Int, Long,,,,, Float UnsignedByte UnsignedShort, Double UnsignedInt, Date UnsignedLong, DateTime, String et Geometry.
  - Le type de vecteur n'est pris en charge dans aucun type.
  - Nested Tous les types ne sont pas pris en charge. Par exemple, `{"value": {"value": "10", "type": "Int"}, "type": "Any"}`.
- Un type de colonne de géométrie est pris en charge dans les colonnes utilisateur. La charge utile de ces colonnes ne doit contenir que des primitives de géométrie de type Point, fournies sous forme de chaînes au format Well-known texte (WKT). Par exemple, POINT (30 10) serait une valeur de géométrie valide.

Exemple de sortie CSV

Compte tenu du fichier CSV suivant :

```
<s3 path>
colA:byte,colB:short,colC:int,colD:long,colE:float,colF:double,colG:string
-128,-32768,-2147483648,-9223372036854775808,1.23456,1.23457,first
127,32767,2147483647,9223372036854775807,nan,nan,second
0,0,0,0,-inf,-inf,third
0,0,0,0,inf,inf,fourth
```

Cet exemple montre le résultat renvoyé par neptune.read à l'aide de la requête suivante :

```
aws neptunedata execute-open-cypher-query \
--open-cypher-query "CALL neptune.read({source: '<s3 path>', format: 'csv'}) YIELD row
 RETURN row" \
--endpoint-url https://my-cluster-name.cluster-abcdefgh1234.us-
east-1.neptune.amazonaws.com:8182
```

```
{
  "results": [{
```

```
      "row": {
        "colD": -9223372036854775808,
        "colC": -2147483648,
        "colE": 1.23456,
        "colB": -32768,
        "colF": 1.2345699999999999,
        "colG": "first",
        "colA": -128
      }
    }, {
      "row": {
        "colD": 9223372036854775807,
        "colC": 2147483647,
        "colE": "NaN",
        "colB": 32767,
        "colF": "NaN",
        "colG": "second",
        "colA": 127
      }
    }, {
      "row": {
        "colD": 0,
        "colC": 0,
        "colE": "-INF",
        "colB": 0,
        "colF": "-INF",
        "colG": "third",
        "colA": 0
      }
    }, {
      "row": {
        "colD": 0,
        "colC": 0,
        "colE": "INF",
        "colB": 0,
        "colF": "INF",
        "colG": "fourth",
        "colA": 0
      }
    }]
}
```

Il n'existe actuellement aucun moyen de définir une étiquette de nœud ou de bordure pour un champ de données provenant d'un fichier CSV. Il est recommandé de partitionner les requêtes en plusieurs requêtes, une pour chacune label/type.

```
CALL neptune.read({source: '<s3 path>', format: 'csv'})
 YIELD row
WHERE row.`~label` = 'airport'
CREATE (n:airport)

CALL neptune.read({source: '<s3 path>', format: 'csv'})
YIELD row
WHERE row.`~label` = 'country'
CREATE (n:country)
```

Gérer les autorisations pour neptune.read ()

Politiques IAM requises

Pour exécuter les requêtes OpenCypher utiliséesneptune.read(), vous devez disposer des autorisations appropriées pour accéder aux données de votre base de données Neptune. Read-only les requêtes nécessitent l'ReadDataViaQueryaction. Les requêtes qui modifient les données nécessitent WriteDataViaQuery des insertions ou DeleteDataViaQuery des suppressions. L'exemple ci-dessous autorise les trois actions sur le cluster spécifié.

En outre, vous devez disposer d'autorisations pour accéder au compartiment S3 contenant vos fichiers de données. La déclaration de politique de Neptunes3Access accorde les autorisations S3 requises :

- **`s3:ListBucket`**: obligatoire pour vérifier l'existence du bucket et le contenu de la liste.
- **`s3:GetObject`**: Nécessaire pour accéder à l'objet spécifié afin que son contenu puisse être lu en vue de son intégration dans les requêtes OpenCypher.

Si votre compartiment S3 utilise le chiffrement côté serveur avec AWS KMS, vous devez également accorder des autorisations KMS. La déclaration de politique Neptunes3KMSAccess permet à Neptune de déchiffrer les données et de générer des clés de données lors de l'accès à des objets S3 chiffrés. Cette condition limite les opérations KMS aux demandes provenant des services S3 et RDS de votre région.

- **`kms:Decrypt`**: Nécessaire pour effectuer le déchiffrement de l'objet chiffré afin que ses données puissent être lues par Neptune.

- **`kms:GenerateDataKey`**: également requis par l'API S3 utilisée pour récupérer les objets à lire.

```
{
  "Sid": "NeptuneQueryAccess",
  "Effect": "Allow",
  "Action": [
      "neptune-db:ReadDataViaQuery",
      "neptune-db:WriteDataViaQuery",
      "neptune-db:DeleteDataViaQuery"
  ],
  "Resource": "arn:aws:neptune-db:<REGION>:<AWS_ACCOUNT_ID>:<CLUSTER_RESOURCE_ID>/*"
},
{
  "Sid": "NeptuneS3Access",
  "Effect": "Allow",
  "Action": [
      "s3:ListBucket",
      "s3:GetObject"
  ],
  "Resource": [
      "arn:aws:s3:::neptune-read-bucket",
      "arn:aws:s3:::neptune-read-bucket/*"
  ]
},
{
  "Sid": "NeptuneS3KMSAccess",
  "Effect": "Allow",
  "Action": [
      "kms:Decrypt",
      "kms:GenerateDataKey"
  ],
  "Resource": "arn:aws:kms:<REGION>:<AWS_ACCOUNT_ID>:key/<KEY_ID>",
  "Condition": {
      "StringEquals": {
        "kms:ViaService": [
            "s3.<REGION>.amazonaws.com",
            "rds.<REGION>.amazonaws.com"
        ]
      }
  }
}
```

Conditions préalables importantes

Ces autorisations et conditions préalables garantissent une intégration sûre et fiable des données S3 dans les requêtes OpenCypher, tout en maintenant des contrôles d'accès et des mesures de protection des données appropriés.

- Authentification IAM : cette fonctionnalité n'est prise en charge que pour les clusters Neptune avec l'authentification IAM activée. Consultez Sécurisation de votre base de données Amazon Neptune pour obtenir des instructions détaillées sur la création et la connexion à des clusters compatibles avec l'authentification IAM.
- Point de terminaison VPC :
  - Un point de terminaison Gateway-type VPC pour Amazon S3 est nécessaire pour permettre à Neptune de communiquer avec Amazon S3.
  - Pour utiliser un AWS KMS chiffrement personnalisé dans la requête, un point de terminaison Interface-type VPC AWS KMS est requis pour permettre à Neptune de communiquer avec. AWS KMS
  - Pour obtenir des instructions détaillées sur la configuration de ce point de terminaison, consultez Création du point de terminaison Amazon S3 VPC.

Données spatiales

Amazon Neptune prend désormais en charge les requêtes spatiales, ce qui vous permet de stocker et d'analyser des données géométriques dans votre graphique. Bien qu'elles soient couramment utilisées pour les emplacements géographiques (comme les coordonnées sur une carte), les entités spatiales fonctionnent avec toutes les données bidimensionnelles où la position et la proximité sont importantes. Utilisez cette fonctionnalité pour répondre à des questions telles que « Quels magasins se trouvent dans un rayon de 8 km de ce client ? » , « Trouvez tous les itinéraires de livraison qui croisent cette zone de service » ou « Quels composants de ce plan d'étage chevauchent la zone CVC ? » Neptune met en œuvre le support spatial à l'aide de fonctions de types spatiaux standard qui fonctionnent avec des points, des polygones et d'autres formes géométriques. Vous pouvez stocker les données spatiales sous forme de propriétés sur les nœuds et les arêtes, puis utiliser des fonctions spatiales pour calculer les distances, vérifier si les points se situent dans les limites ou rechercher des régions qui se chevauchent, le tout dans vos requêtes OpenCypher.

Cas d'utilisation courants :

- Applications géographiques : Location-based recommandations, géofencing, planification d'itinéraires et analyse du territoire
- Gestion des installations et de l'espace : disposition du plan d'étage, placement de l'équipement et couverture des zones
- Topologie du réseau : cartographie de l'infrastructure physique, zones de couverture et limites de service
- Conception et CAO : positionnement des composants, détection des collisions et relations spatiales dans les conceptions 2D
- Développement du jeu : positionnement des personnages, détection des collisions et calculs de la zone d'effet

L'implémentation des types spatiaux dans Amazon Neptune suit les directives ISO/IEC 13249-3:2016, comme les autres bases de données. Ils Fonctions spatiales sont disponibles dans le langage de requête OpenCypher.

Système de coordonnées

Neptune possède un identifiant de référence spatiale (SRID) pour l'ensemble d'une base de données. L'homogénéité du système de coordonnées réduit les erreurs des utilisateurs lors des requêtes et améliore les performances de la base de données. La première version (1.4.7.0) prend en charge le système de coordonnées cartésien, également appelé SRID 0.

L'implémentation Neptune du SRID 0 est compatible avec les valeurs de longitude et de latitude. `ST_DistanceSpheroid`À utiliser pour calculer les distances en fonction de WGS84/SRID 4326.

L'implémentation actuelle prend en charge le stockage de coordonnées tridimensionnelles. Les fonctions spatiales ne prennent actuellement en charge que l'utilisation des coordonnées des axes x et y (bidimensionnelles). Les coordonnées de l'axe Z ne sont actuellement pas prises en charge par les fonctions spatiales disponibles.

Stockage des données de localisation

Stockez les données de localisation sur les nœuds et les arêtes à l'aide du type de propriété Geometry. Créez des valeurs géométriques à partir du format Well-Known texte (WKT), une méthode standard pour représenter des formes géographiques sous forme de texte. Par exemple, pour enregistrer l'emplacement d'un point :

```
CREATE (n:airport {code: 'ATL', location: ST_GeomFromText('POINT (-84.4281 33.6367)')})
```

Lorsque vous travaillez avec des coordonnées géographiques, le premier argument (x) représente la longitude et le second argument (y) représente la latitude. Cela suit l'ordre de coordonnées standard utilisé dans les bases de données spatiales et la norme ISO 19125.

> **Note**
>
> Neptune prend désormais en charge un nouveau type de données appelé « Géométrie ». La propriété de géométrie d'un nœud ou d'une arête peut être créée à partir d'une chaîne WKT à l'aide de la `ST_GeomFromText` fonction.
> Neptune stockera automatiquement les données relatives aux points dans un index spatial spécialisé afin d'améliorer les performances des fonctions de types spatiaux. Par exemple, l'index spatial spécialisé accélère le processus `ST_Contains` utilisé pour trouver les points dans un polygone.
> Page Wikipedia pour la représentation Well-Known textuelle de la géométrie

Chargement de données spatiales en masse

Lorsque vous chargez des données en bloc, spécifiez le type de géométrie dans votre en-tête CSV. Neptune analysera les chaînes WKT et créera les propriétés de géométrie appropriées :

```
:ID,:LABEL,code:String,city:String,location:Geometry
21,airport,ATL,Atlanta,POINT (-84.42810059 33.63669968)
32,airport,ANC,Anchorage,POINT (-149.9960022 61.17440033)
43,airport,AUS,Austin,POINT (-97.66989899 30.19449997)
```

Pour plus de détails sur le format CSV, voir Format de chargement groupé OpenCypher.

Interrogation des données spatiales

Les exemples de requêtes suivants utilisent le jeu de données des routes aériennes pour montrer comment utiliser les fonctions spatiales dans Neptune.

Si vos données possèdent des propriétés de latitude et de longitude distinctes au lieu d'une propriété de géométrie, vous pouvez les convertir en points au moment de la requête. Trouvez les 10 aéroports les plus proches d'un lieu donné :

```
MATCH (a:airport)
WITH a, ST_GeomFromText('POINT (' + a.lon + ' ' + a.lat + ')') AS airportLocation
```

```
WITH a, airportLocation, ST_Distance(ST_GeomFromText('POINT (-84.4281 33.6367)'),
 airportLocation) AS distance
WHERE distance IS NOT NULL
RETURN a.code, a.city, distance
ORDER BY distance ASC
LIMIT 10
```

Si vous avez déjà enregistré des emplacements`ST_Point`, vous pouvez utiliser ces valeurs de position directement :

1. Définition de la propriété

```
MATCH (a:airport)
SET a.location = ST_GeomFromText('POINT (' + a.lon + ' ' + a.lat + ')')
```

2. Requête utilisant ST_Distance :

```
MATCH (a:airport)
WHERE a.location IS NOT NULL
WITH a, ST_Distance(ST_GeomFromText('POINT (-84.4281 33.6367)'), a.location) AS
 distance
RETURN a.code, a.city, distance
ORDER BY distance ASC
LIMIT 10
```

Utilisation du pilote Bolt

La plupart des méthodes de requête renvoient les valeurs de géométrie sous forme de chaînes WKT, lisibles par l'homme. Si vous utilisez le pilote Bolt, les valeurs de géométrie sont renvoyées au format WKB (Well-Known binaire) pour plus d'efficacité. Convertissez le WKB en objet de géométrie dans votre application :

```
try (Session session = driver.session()) {
    Result result = session.run("MATCH (n:airport {code: 'ATL'}) RETURN n.location as
 geom");

    Record record = result.single();
    byte[] wkbBytes = record.get("geom").asByteArray();

    // Convert WKB to Geometry object using JTS library
    WKBReader wkbReader = new WKBReader();
```

```
    Geometry geom = wkbReader.read(wkbBytes);
}
```

Fonctions spatiales

Les fonctions spatiales suivantes sont disponibles dans Neptune OpenCypher pour travailler avec des types de données géométriques :

- ST_Point
- ST_ GeomFromText
- ST_ AsText
- ST_ GeometryType
- ST_Equals
- ST_Contains
- ST_Intersects
- ST_Distance
- ST_ DistanceSpheroid
- ST_Enveloppe
- ST_Buffer

ST_Point

ST_Point renvoie un point à partir des valeurs de coordonnées en entrée.

Syntaxe

```
ST_Point(x, y, z)
```

Arguments

- x- Une valeur de type de données DOUBLE PRECISION qui représente une première coordonnée.
- y- Une valeur de type de données DOUBLE PRECISION qui représente une deuxième coordonnée.
- z- (facultatif)

Ordre de coordonnées

Lorsque vous travaillez avec des coordonnées géographiques, le premier argument (x) représente la longitude et le second (y) représente la latitude. Cela suit l'ordre de coordonnées standard utilisé dans les bases de données spatiales et la norme ISO 19125.

```
// Correct: longitude first, latitude second
ST_Point(-84.4281, 33.6367)  // Atlanta airport

// Incorrect: latitude first, longitude second
ST_Point(33.6367, -84.4281)  // This will return NaN in distance calculations
```

Plages de coordonnées valides

Pour les données géographiques, assurez-vous que les coordonnées se situent dans des plages valides :

- Longitude (x) : -180 à 180
- Latitude (y) : -90 à 90

Les coordonnées situées en dehors de ces plages seront renvoyées NaN (ce n'est pas un nombre) lorsqu'elles sont utilisées avec des fonctions de calcul de distance telles queST_DistanceSpheroid.

Type de retour

GÉOMÉTRIE du sous-type POINT

Si x ou y est null, null est renvoyé.

Exemples

Ce qui suit construit une géométrie ponctuelle à partir des coordonnées en entrée.

```
RETURN ST_Point(5.0, 7.0);
POINT(5 7)
```

ST_ GeomFromText

ST_ GeomFromText construit un objet géométrique à partir d'une représentation textuelle connue (WKT) d'une géométrie d'entrée.

Syntaxe

```
ST_GeomFromText(wkt_string)
```

Arguments

- `wkt_string`- Une valeur de type de données STRING qui est une représentation WKT d'une géométrie.

Type de retour

GEOMETRY

Si wkt_string est null, null est renvoyé.

Si wkt_string n'est pas valide, alors a BadRequestException est renvoyé.

Exemples

```
RETURN ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))')
POLYGON((0 0,0 1,1 1,1 0,0 0))
```

ST_ AsText

ST_ AsText renvoie la représentation textuelle connue (WKT) d'une géométrie d'entrée.

Syntaxe

```
ST_AsText(geo)
```

Arguments

- `geo`- Une valeur de type de données GEOMETRY, ou une expression qui correspond à une GEOMETRY.

Type de retour

CHAÎNE

Si geo est null, null est renvoyé.

Si le paramètre d'entrée n'est pas une géométrie, la valeur a BadRequestException est renvoyée.

Si le résultat est supérieur à une CHAÎNE de 64 Ko, une erreur est renvoyée.

Exemples

```
RETURN ST_AsText(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'))
POLYGON((0 0,0 1,1 1,1 0,0 0))
```

ST_ GeometryType

ST_ GeometryType renvoie le type de géométrie sous forme de chaîne.

Syntaxe

```
ST_GeometryType(geom)
```

Arguments

- `geom`- Une valeur du type de données GEOMETRY ou une expression qui correspond à un type GEOMETRY.

Type de retour

CHAÎNE

Si geom est null, null est renvoyé.

Si le paramètre d'entrée n'est pas une géométrie, la valeur a BadRequestException est renvoyée.

Exemples

```
RETURN ST_GeometryType(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'));
ST_LineString
```

ST_Equals

ST_equals renvoie la valeur true si les projections 2D des géométries en entrée sont topologiquement égales. Les géométries sont considérées comme topologiquement égales si elles comportent des

ensembles de points égaux. Dans des géométries topologiquement égales, l'ordre des sommets peut différer tout en maintenant cette égalité.

Syntaxe

```
ST_Equals(geom1, geom2)
```

Arguments

- geom1- Une valeur du type de données GEOMETRY ou une expression qui correspond à un type GEOMETRY.
- geom2- Une valeur du type de données GEOMETRY ou une expression qui correspond à un type GEOMETRY. Cette valeur est comparée à geom1 afin de déterminer si elle est égale à geom1.

Type de retour

BOOLEAN

Si geom1 ou geom2 est null, null est renvoyé.

Si geom1 ou geom2 ne sont pas des géométries, alors a est renvoyé. BadRequestException

Exemples

```
RETURN ST_Equals(
    ST_GeomFromText('POLYGON ((0 2,1 1,0 -1,0 2))'),
    ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
false
```

Ce qui suit vérifie si les deux chaînes de lignes sont géométriquement égales.

```
RETURN ST_Equals(
    ST_GeomFromText('LINESTRING (1 0, 10 0)'),
    ST_GeomFromText('LINESTRING(1 0,5 0,10 0)'));
true
```

ST_Contains

ST_Contains renvoie true si la projection 2D de la première géométrie en entrée contient la projection 2D de la deuxième géométrie en entrée. La géométrie A contient la géométrie B si chaque

point de B est un point de A et que leur intérieur comporte une intersection non vide. ST_Contains (A, B) est équivalent à ST_Within (B, A).

Syntaxe

```
ST_Contains(geom1, geom2)
```

Arguments

- geom1- Une valeur de type GEOMETRY ou une expression qui correspond à un type GEOMETRY.
- geom2- Une valeur de type GEOMETRY ou une expression qui correspond à un type GEOMETRY. Cette valeur est comparée à geom1 afin de déterminer si elle est contenue dans geom1.

Type de retour

BOOLEAN

Si geom1 ou geom2 est null, null est renvoyé.

Si le paramètre d'entrée n'est pas une géométrie, la valeur a BadRequestException est renvoyée.

Exemples

```
RETURN ST_Contains(
    ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
    ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
false
```

ST_Intersects

ST_Intersects renvoie true si les projections 2D des deux géométries d'entrée ont au moins un point en commun.

Syntaxe

```
ST_Intersects(geom1, geom2)
```

Arguments

- geom1- Une valeur du type de données GEOMETRY ou une expression qui correspond à un type GEOMETRY.

- `geom2`- Une valeur du type de données GEOMETRY ou une expression qui correspond à un type GEOMETRY.

Type de retour

BOOLEAN

Si geom1 ou geom2 est null, null est renvoyé.

Si le paramètre d'entrée n'est pas une géométrie, la valeur a BadRequestException est renvoyée.

Exemples

```
RETURN ST_Intersects(
    ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(2 2,2 5,5 5,5 2,2 2))'),
    ST_GeomFromText('MULTIPOINT((4 4),(6 6))'));
true
```

ST_Distance

Pour les géométries d'entrée, ST_Distance renvoie la distance euclidienne minimale entre les projections 2D des deux valeurs géométriques d'entrée.

Syntaxe

```
ST_Distance(geo1, geo2)
```

Arguments

- `geo1`- Une valeur du type de données GEOMETRY, ou une expression qui correspond à un type GEOMETRY.
- `geo2`- Une valeur de type de données GEOMETRY, ou une expression qui correspond à une GEOMETRY.

Type de retour

DOUBLE PRÉCISION dans les mêmes unités que les géométries d'entrée.

Si geo1 ou geo2 est nul, nul est renvoyé.

Si le paramètre d'entrée n'est pas une géométrie, la valeur a BadRequestException est renvoyée.

Exemples

```
RETURN ST_Distance(
    ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
    ST_GeomFromText('POLYGON((-1 -3,-2 -1,0 -3,-1 -3))'));
1.4142135623731
```

ST_ DistanceSpheroid

Renvoie la distance minimale en mètres entre deux lon/lat géométries. Le sphéroïde est WGS84/ SRID 4326.

Syntaxe

```
ST_DistanceSpheroid(geom1, geom2);
```

Arguments

- geom1- Une valeur du type de données GEOMETRY ou une expression qui correspond à un type GEOMETRY.
- geom2- Une valeur du type de données GEOMETRY ou une expression qui correspond à un type GEOMETRY.

Type de retour

FLOAT

Si geom est null, null est renvoyé.

Exemples

```
RETURN ST_DistanceSpheroid(
    ST_GeomFromText('POINT(-110 42)'),
    ST_GeomFromText('POINT(-118 38)'))
814278.77
```

ST_Enveloppe

ST_Envelope renvoie le cadre de délimitation minimal de la géométrie en entrée, comme suit :

- Si la géométrie en entrée est vide, la géométrie renvoyée sera POINT EMPTY.
- Si le cadre de délimitation minimal de la géométrie en entrée dégénère en un point, la géométrie renvoyée est un point.
- Si aucune des réponses précédentes n'est vraie, la fonction renvoie un polygone orienté dans le sens inverse des aiguilles d'une montre dont les sommets sont les coins du cadre de délimitation minimal.

Pour toutes les entrées non vides, la fonction fonctionne sur la projection 2D de la géométrie en entrée.

Syntaxe

```
ST_Envelope(geom)
```

Arguments

- `geom`- Une valeur du type de données GEOMETRY ou une expression qui correspond à un type GEOMETRY.

Type de retour

GEOMETRY

Si geom est null, null est renvoyé.

Exemples

```
RETURN ST_Envelope(ST_GeomFromText("POLYGON ((2 1, 4 3, 6 1, 5 5, 3 4, 2 1))"))
POLYGON ((2 1, 6 1, 6 5, 2 5, 2 1))
```

ST_Buffer

ST_Buffer renvoie une géométrie 2D qui représente tous les points dont la distance par rapport à la géométrie d'entrée projetée sur le plan cartésien XY est inférieure ou égale à la distance d'entrée.

Syntaxe

```
ST_Buffer(geom, distance, number_of_segments_per_quarter_circle)
```

Arguments

- `geom`- Une valeur du type de données GEOMETRY ou une expression qui correspond à un type GEOMETRY.
- `distance`- Une valeur de type de données DOUBLE PRECISION qui représente la distance (ou le rayon) de la mémoire tampon.
- `number_of_segments_per_quarter_circle`- Une valeur de type INTEGER (doit être supérieure ou égale à 0). Cette valeur détermine le nombre de points à approcher d'un quart de cercle autour de chaque sommet de la géométrie d'entrée. Les valeurs négatives ont pour valeur par défaut zéro. La valeur par défaut est de 8.

Type de retour

GEOMETRY

La fonction ST_Buffer renvoie une géométrie bidimensionnelle (2D) dans le plan cartésien XY.

Exemples

```
RETURN ST_Buffer(ST_GeomFromText('LINESTRING (1 2,5 2,5 8)'), 2, 4);
POLYGON ((3 4, 3 8, 3.1522409349774265 8.76536686473018,
         3.585786437626905 9.414213562373096, 4.234633135269821 9.847759065022574,
         5 10, 5.765366864730179 9.847759065022574,
         6.414213562373095 9.414213562373096, 6.847759065022574 8.76536686473018,
         7 8, 7 2, 6.847759065022574 1.2346331352698203,
         6.414213562373095 0.5857864376269051, 5.765366864730179 0.1522409349774265,
         5 0, 1 0, 0.2346331352698193 0.152240934977427,
         -0.4142135623730954 0.5857864376269051,
         -0.8477590650225737 1.2346331352698208, -1 2.0000000000000004,
         -0.8477590650225735 2.7653668647301797,
         -0.4142135623730949 3.414213562373095,
         0.2346331352698206 3.8477590650225735, 1 4, 3 4))
```

Ce qui suit renvoie la zone tampon de la géométrie du point d'entrée qui correspond approximativement à un cercle. Étant donné que la commande spécifie 3 comme nombre de segments par quart de cercle, la fonction utilise trois segments pour approcher le quart de cercle.

```
RETURN ST_Buffer(ST_GeomFromText('POINT (1 1)'), 1.0, 8));
POLYGON ((2 1, 1.9807852804032304 0.8049096779838718,
```

```
 1.9238795325112867 0.6173165676349102, 1.8314696123025453 0.4444297669803978,
 1.7071067811865475 0.2928932188134525, 1.5555702330196022 0.1685303876974548,
 1.3826834323650898 0.0761204674887133, 1.1950903220161284 0.0192147195967696,
 1 0, 0.8049096779838718 0.0192147195967696, 0.6173165676349103 0.0761204674887133,
0.444429766980398 0.1685303876974545, 0.2928932188134525 0.2928932188134524,
 0.1685303876974546 0.4444297669803978, 0.0761204674887133 0.6173165676349102,
 0.0192147195967696 0.8049096779838714, 0 0.9999999999999999,
 0.0192147195967696 1.1950903220161284, 0.0761204674887132 1.3826834323650896,
 0.1685303876974545 1.555570233019602, 0.2928932188134523 1.7071067811865475,
 0.4444297669803978 1.8314696123025453, 0.6173165676349097 1.9238795325112865,
 0.8049096779838714 1.9807852804032304, 0.9999999999999998 2,
 1.1950903220161284 1.9807852804032304, 1.38268343236509 1.9238795325112865,
 1.5555702330196017 1.8314696123025453, 1.7071067811865475 1.7071067811865477,
 1.8314696123025453 1.5555702330196022, 1.9238795325112865 1.3826834323650905,
 1.9807852804032304 1.1950903220161286, 2 1))
```

Accès au graphe Neptune avec SPARQL

SPARQL est un langage de requête pour RDF (Resource Description Framework), qui est un format de données de graphe conçu pour le web. Amazon Neptune est compatible avec SPARQL 1.1. En d'autres termes, vous pouvez vous connecter à une instance de base de données Neptune et interroger le graphe à l'aide du langage de requête décrit dans la spécification SPARQL 1.1 Query Language.

Une requête dans SPARQL se compose d'une clause `SELECT` pour spécifier les variables à renvoyer et une clause `WHERE` clause pour spécifier les données de correspondance du graphe. Si vous ne connaissez pas les requêtes SPARQL, consultez Writing Simple Queries dans la section SPARQL 1.1 Query Language.

> **Important**
>
> Pour charger des données, `SPARQL UPDATE INSERT` peut fonctionner correctement pour un petit ensemble de données, mais si vous avez besoin de charger une quantité importante de données à partir d'un fichier, consultez Utilisation du chargeur en vrac Amazon Neptune pour ingérer des données.

Pour plus d'informations sur les spécificités de l'implémentation SPARQL dans Neptune, consultez Conformité avec les normes SPARQL.

Avant de commencer, les prérequis suivants doivent être remplis :

- Vous devez disposer d'une instance de base de données Neptune. Pour plus d'informations sur la création d'une instance de base de données Neptune, consultez Création d'un cluster Amazon Neptune.
- Vous devez disposer d'une instance Amazon EC2 dans le même cloud privé virtuel (VPC) que l'instance de base de données Neptune.

Rubriques

- Utilisation de la console RDF4J pour se connecter à une instance de base de données Neptune
- Utilisation de RDF4J Workbench pour se connecter à une instance de base de données Neptune
- Utilisation de Java pour se connecter à une instance de base de données Neptune
- API HTTP SPARQL
- Indicateurs de requête SPARQL
- Comportement de SPARQL DESCRIBE par rapport au graphe par défaut
- API de statut des requêtes SPARQL
- Annulation de requêtes SPARQL
- Utilisation du protocole HTTP SPARQL 1.1 Graph Store (GSP) dans Amazon Neptune
- Analyse de l'exécution des requêtes Neptune à l'aide de SPARQL : explication
- Requêtes fédérées SPARQL dans Neptune à l'aide de l'extension SERVICE

Utilisation de la console RDF4J pour se connecter à une instance de base de données Neptune

La console RDF4J vous permet de tester les graphes et les requêtes RDF (Resource Description Framework) dans un environnement REPL (Read-Eval-Print Loop).

Vous pouvez ajouter une base de données orientée graphe en tant que référentiel et l'interroger à partir de la console RDF4J. Cette section vous accompagne dans la configuration de la console RDF4J pour une connexion à distance à l'instance de base de données Neptune.

Pour se connecter à Neptune à l'aide de la console RDF4J

1. Téléchargez le kit SDK RDF4J à partir de la page de téléchargement sur le site web RDF4J.

2. Décompressez le fichier RDF4J SDK.
3. Dans un terminal, accédez au répertoire RDF4J SDK, puis saisissez la commande suivante pour exécuter la console RDF4J :

```
bin/console.sh
```

Vous devez voir des résultats similaires à ce qui suit :

```
14:11:51.126 [main] DEBUG o.e.r.c.platform.PlatformFactory - os.name = linux
14:11:51.130 [main] DEBUG o.e.r.c.platform.PlatformFactory - Detected Posix
 platform
Connected to default data directory
RDF4J Console 3.6.1

3.6.1
Type 'help' for help.
>
```

Vous êtes maintenant à l'invite >. Il s'agit de l'invite générale de la console RDF4J. Vous utilisez cette invite pour configurer les référentiels et autres opérations. Un référentiel possède sa propre invite pour l'exécution des requêtes.

4. Saisissez ce qui suit à l'invite > afin de créer un répertoire SPARQL pour votre instance de base de données Neptune :

```
create sparql
```

5. La console RDF4J Console vous invite à fournir les valeurs des variables requises pour se connecter au point de terminaison SPARQL.

```
Please specify values for the following variables:
```

Indiquez l'une des valeurs suivantes :

Nom de variable	Value
Point de terminaison de requête SPARQL	`https://your-neptune-endpoint :port/sparql`

Point de terminaison de mise à jour SPARQL	`https://your-neptune-endpoint :port/sparql`
ID de référentiel local [endpoint@localhost]	`neptune`
Titre de référentiel [référentiel de point de terminaison SPARQL @localhost]	`Neptune DB instance`

Consultez la section Connexion aux points de terminaison Amazon Neptune pour découvrir comment trouver l'adresse de votre instance de base de données Neptune.

Si l'opération est réussie, vous voyez le message suivant :

```
Repository created
```

6. À l'invite de commande >, entrez ce qui suit pour vous connecter à l'instance de base de données Neptune.

```
open neptune
```

Si l'opération est réussie, vous voyez le message suivant :

```
Opened repository 'neptune'
```

Vous êtes maintenant à l'invite `neptune>`. À l'invite de commande, vous pouvez exécuter des requêtes par rapport au graphe Neptune.

> **Note**
>
> Maintenant que vous avez ajouté le répertoire, la prochaine fois que vous exécuterez `bin/console.sh`, vous pourrez immédiatement exécuter la commande `open neptune` pour vous connecter à l'instance de base de données Neptune.

7. À l'invite `neptune>`, saisissez la commande suivante pour exécuter une requête SPARQL qui renvoie 10 triplets au plus (sujet-prédicat-objet) dans le graphe à l'aide de la requête `?s ?p ?`

o avec une limite égale à 10. Pour interroger quelque chose d'autre, remplacez le texte après la commande `sparql` par une autre requête SPARQL.

```
sparql select ?s ?p ?o where {?s ?p ?o} limit 10
```

Utilisation de RDF4J Workbench pour se connecter à une instance de base de données Neptune

Cette section vous accompagne lors la connexion à une l'instance de base de données Neptune avec RDF4J Workbench et RDF4J Server. RDF4J Server est obligatoire, car il agit comme proxy entre le point de terminaison HTTP REST SPARQL Neptune et RDF4J Workbench.

RDF4J Workbench fournit une interface simple pour être utilisé avec un graphe, y compris le chargement des fichiers locaux. Pour de plus amples informations, veuillez consulter la section Add dans la documentation RDF4J.

Conditions préalables

Avant de commencer, vous devez exécuter les actions suivantes :

- Installez Java 1.8 ou version ultérieure.
- Installez RDF4J Server et RDF4J Workbench. Pour de plus amples informations, veuillez consulter Installing RDF4J Server and RDF4J Workbench.

Pour utiliser RDF4J Workbench pour se connecter à Neptune

1. Dans un navigateur web, accédez à l'URL où l'application web RDF4J Workbench est déployée. Par exemple, si vous utilisez Apache Tomcat, l'URL est : https ://*ec2_hostname*: 8080/rdf4j - workbench/.
2. Si vous êtes invité à vous connecter à RDF4J Server, vérifiez que RDF4J Server est installé, en cours d'exécution et que l'URL du serveur est correct. Passez ensuite à l'étape suivante.
3. Dans le volet de gauche, sélectionnez Nouveau référentiel.

   Dans Nouveau référentiel :

   - Dans la liste déroulante Type, choisissez Proxy de point de terminaison SPARQL.
   - Pour ID, saisissez neptune.

- Pour Titre, tapez Instance de base de données Neptune.

Choisissez Suivant.

4. Dans Nouveau référentiel :

   - Pour SPARQL query endpoint URL (URL de point de terminaison de requête SPARQL), entrez `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.
   - Pour SPARQL update endpoint URL (URL de point de terminaison de mise à jour SPARQL), entrez `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.

   Consultez la section Connexion aux points de terminaison Amazon Neptune pour découvrir comment trouver l'adresse de votre instance de base de données Neptune.

   Choisissez Créer.

5. Le référentiel neptune figure désormais dans la liste des référentiels. Quelques minutes peuvent s'écouler avant que vous ne puissiez utiliser le nouveau référentiel.
6. Dans la colonne Id de la table, choisissez le lien neptune.
7. Dans le volet de gauche, choisissez Requête.

   > **Note**
   >
   > Si les options de menu sous Explorer sont désactivées, vous devez peut-être vous reconnecter au serveur RDF4J et choisir à nouveau le référentiel neptune.
   > Pour ce faire, vous pouvez utiliser les liens [modifier] dans le coin supérieur droit.

8. Dans le champ de requête, tapez la requête SPARQL suivante, puis choisissez Exécuter.

```
select ?s ?p ?o where {?s ?p ?o} limit 10
```

L'exemple précédent renvoie 10 triplets au plus (sujet-prédicat-objet) dans le graphe à l'aide de la requête `?s ?p ?o` avec une limite égale à 10.

Utilisation de Java pour se connecter à une instance de base de données Neptune

Cette section vous accompagne lors de l'exécution d'un exemple Java complet qui se connecte à une l'instance de base de données Amazon Neptune et exécute une requête SPARQL.

Vous devez suivre ces instructions à partir d'une instance Amazon EC2 dans le même cloud privé virtuel (VPC) (VPC) que l'instance de base de données Neptune.

Pour se connecter à Neptune à l'aide de Java

1. Installez Apache Maven sur votre instance EC2. Si vous utilisez Amazon Linux 2023 (de préférence), utilisez :

   ```
   sudo dnf update -y
   sudo dnf install maven -y
   ```

   Si vous utilisez Amazon Linux 2, téléchargez le dernier fichier binaire à l'adresse suivante https://maven.apache.org/download.cgi:

   ```
   sudo yum remove maven -y
   wget https://dlcdn.apache.org/maven/maven-3/ <version>/binaries/apache-maven-
   <version>-bin.tar.gz
   sudo tar -xzf apache-maven-<version>-bin.tar.gz -C /opt/
   sudo ln -sf /opt/apache-maven-<version> /opt/maven
   echo 'export MAVEN_HOME=/opt/maven' >> ~/.bashrc
   echo 'export PATH=$MAVEN_HOME/bin:$PATH' >> ~/.bashrc
   source ~/.bashrc
   ```

2. Cet exemple a été testé avec Java 8 uniquement. Entrez ce qui suit pour installer Java 8 sur votre instance EC2 :

   ```
   sudo yum install java-1.8.0-devel
   ```

3. Entrez la commande suivante pour définir Java 8 en tant qu'exécution par défaut sur votre instance EC2 :

   ```
   sudo /usr/sbin/alternatives --config java
   ```

   Lorsque vous y êtes invité, saisissez le nombre pour Java 8.

4. Entrez la commande suivante pour définir Java 8 en tant que compilateur par défaut de votre instance EC2 :

```
sudo /usr/sbin/alternatives --config javac
```

   Lorsque vous y êtes invité, saisissez le nombre pour Java 8.

5. Dans un nouveau répertoire , créez un fichier `pom.xml`, puis ouvrez-le dans un éditeur de texte.

6. Copiez ce qui suit dans le fichier `pom.xml` et enregistrez-le (vous pouvez normalement remplacer les numéros de version par la dernière version stable) :

```
<project xmlns="https://maven.apache.org/POM/4.0.0" xmlns:xsi="https://
www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.amazonaws</groupId>
  <artifactId>RDFExample</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>RDFExample</name>
  <url>https://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>org.eclipse.rdf4j</groupId>
      <artifactId>rdf4j-runtime</artifactId>
      <version>3.6</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>exec-maven-plugin</artifactId>
          <version>1.2.1</version>
          <configuration>
            <mainClass>com.amazonaws.App</mainClass>
          </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
```

```
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

> **Note**
>
> Si vous modifiez un projet Maven existant, la dépendance obligatoire est mise en évidence dans le code précédent.

7. Pour créer des sous-répertoires pour l'exemple de code source (`src/main/java/com/amazonaws/`), saisissez ce qui suit sur la ligne de commande :

```
mkdir -p src/main/java/com/amazonaws/
```

8. Dans le répertoire `src/main/java/com/amazonaws/`, créez un fichier `App.java`, puis ouvrez-le dans un éditeur de texte.
9. Copiez ce qui suit dans le fichier `App.java`. Remplacez *`your-neptune-endpoint`* par l'adresse de votre instance de base de données Neptune.

> **Note**
>
> Consultez la section Connexion aux points de terminaison Amazon Neptune pour découvrir comment trouver le nom d'hôte de votre instance de base de données Neptune.

```
package com.amazonaws;

import org.eclipse.rdf4j.repository.Repository;
import org.eclipse.rdf4j.repository.http.HTTPRepository;
import org.eclipse.rdf4j.repository.sparql.SPARQLRepository;

import java.util.List;
import org.eclipse.rdf4j.RDF4JException;
```

```
import org.eclipse.rdf4j.repository.RepositoryConnection;
import org.eclipse.rdf4j.query.TupleQuery;
import org.eclipse.rdf4j.query.TupleQueryResult;
import org.eclipse.rdf4j.query.BindingSet;
import org.eclipse.rdf4j.query.QueryLanguage;
import org.eclipse.rdf4j.model.Value;

public class App
{
    public static void main( String[] args )
    {
        String sparqlEndpoint = "https://your-neptune-endpoint:port/sparql";
        Repository repo = new SPARQLRepository(sparqlEndpoint);
        repo.initialize();

        try (RepositoryConnection conn = repo.getConnection()) {
           String queryString = "SELECT ?s ?p ?o WHERE { ?s ?p ?o } limit 10";

           TupleQuery tupleQuery = conn.prepareTupleQuery(QueryLanguage.SPARQL,
 queryString);

           try (TupleQueryResult result = tupleQuery.evaluate()) {
              while (result.hasNext()) {  // iterate over the result
                   BindingSet bindingSet = result.next();

                   Value s = bindingSet.getValue("s");
                   Value p = bindingSet.getValue("p");
                   Value o = bindingSet.getValue("o");

                   System.out.print(s);
                   System.out.print("\t");
                   System.out.print(p);
                   System.out.print("\t");
                   System.out.println(o);
              }
           }
        }
    }
}
```

10. Compilez et exécutez l'exemple à l'aide de la commande Maven suivante :

```
mvn compile exec:java
```

L'exemple précédent renvoie 10 triplets au plus (sujet-prédicat-objet) dans le graphe à l'aide de la requête `?s ?p ?o` avec une limite égale à 10. Pour interroger autre chose, remplacez la requête par une autre requête SPARQL.

L'itération des résultats dans l'exemple imprime la valeur de chaque variable renvoyée. L'objet `Value` est convertie en une `String`, puis imprimé. Si vous modifiez la partie `SELECT` de la requête, vous devez modifier le code.

API HTTP SPARQL

Les requêtes HTTP SPARQL sont acceptées au point de terminaison suivant : `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`

Pour plus d'informations sur la connexion à Amazon Neptune avec SPARQL, consultez Accès au graphe Neptune avec SPARQL.

Pour plus d'informations sur le langage de requête et le protocole SPARQL, consultez les spécifications SPARQL 1.1 Protocol et SPARQL 1.1 Query Language.

Les rubriques suivantes fournissent des informations sur les formats de sérialisation RDF SPARQL et la façon d'utiliser l'API HTTP SPARQL avec Neptune.

Table des matières

- Utilisation du point de terminaison HTTP REST pour se connecter à une instance de base de données Neptune
- En-têtes de suivi HTTP facultatifs pour les réponses SPARQL en plusieurs parties
- Types de médias RDF utilisés par SPARQL dans Neptune
  - Formats de sérialisation RDF utilisés par Neptune SPARQL
  - Formats de sérialisation des résultats SPARQL utilisés par Neptune SPARQL
  - Media-Types que Neptune peut utiliser pour importer des données RDF
  - Media-Types que Neptune peut utiliser pour exporter les résultats des requêtes
- Utilisation de SPARQL UPDATE LOAD pour importer des données dans Neptune
- Utilisation de SPARQL UPDATE UNLOAD pour supprimer des données de Neptune

Utilisation du point de terminaison HTTP REST pour se connecter à une instance de base de données Neptune

> **Note**
>
> Neptune ne prend actuellement pas en charge HTTP/2 les requêtes d'API REST. Les clients doivent l'utiliser HTTP/1.1 lorsqu'ils se connectent aux points de terminaison.

Les instructions suivantes vous guident pour la connexion au point de terminaison SPARQL à l'aide de la commande curl, avec une connexions via HTTPS et en utilisant la syntaxe HTTP. Vous devez suivre ces instructions à partir d'une instance Amazon EC2 dans le même cloud privé virtuel (VPC) (VPC) que l'instance de base de données Neptune.

Le point de terminaison HTTP pour les requêtes SPARQL dans une instance de base de données Neptune est : `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.

> **Note**
>
> Consultez la section Connexion aux points de terminaison Amazon Neptune pour découvrir comment trouver le nom d'hôte de votre instance de base de données Neptune.

Amazon Neptune fournit un point de terminaison HTTP pour les requêtes SPARQL. L'interface REST est compatible avec SPARQL version 1.1.

REQUÊTE avec HTTP POST

L'exemple suivant utilise curl pour soumettre un **`QUERY`** SPARQL via HTTP POST.

```
curl -X POST --data-binary 'query=select ?s ?p ?o where {?s ?p ?o} limit 10'
 https://your-neptune-endpoint:port/sparql
```

L'exemple précédent renvoie 10 triplets au plus (sujet-prédicat-objet) dans le graphe à l'aide de la requête `?s ?p ?o` avec une limite égale à 10. Pour interroger autre chose, remplacez la requête par une autre requête SPARQL.

> **Note**
>
> Le type de support MIME par défaut d'une réponse est `application/sparql-results+json` pour les requêtes `SELECT` et `ASK`.
> Le type MIME par défaut d'une réponse est `application/n-quads` pour les requêtes `CONSTRUCT` et `DESCRIBE`.
> Pour obtenir la liste des types de médias utilisés par Neptune pour la sérialisation, consultez Formats de sérialisation RDF utilisés par Neptune SPARQL.

MISE À JOUR à l'aide de HTTP POST

L'exemple suivant utilise curl pour soumettre un **`UPDATE`** SPARQL via HTTP POST.

```
curl -X POST --data-binary 'update=INSERT DATA { <https://test.com/s> <https://
test.com/p> <https://test.com/o> . }' https://your-neptune-endpoint:port/sparql
```

L'exemple précédent insère le triplet suivant dans le graphe SPARQL par défaut : `<https://test.com/s> <https://test.com/p> <https://test.com/o>`

En-têtes de suivi HTTP facultatifs pour les réponses SPARQL en plusieurs parties

La réponse HTTP aux requêtes et aux mises à jour SPARQL est souvent renvoyée en plusieurs parties ou fragments. Il peut être difficile de diagnostiquer un échec qui survient après qu'une requête ou une mise à jour commence à envoyer ces fragments, d'autant plus que le premier arrive avec le code de statut HTTP `200`.

À moins que vous ne demandiez explicitement des en-têtes de suivi, Neptune ne signale un tel échec qu'en ajoutant un message d'erreur dans le corps du message, qui est généralement corrompu.

Pour faciliter la détection et le diagnostic de ce type de problème, vous pouvez inclure un en-tête de suivi à encodage de transfert (TE), `te: trailers`, dans votre demande (voir, par exemple, la page MDN sur les en-têtes de requête TE). De cette manière, Neptune inclura deux nouveaux champs d'en-tête dans les en-têtes de suivi des fragments de réponse :

- `X-Neptune-Status` : contient le code de réponse suivi d'un nom court. Par exemple, en cas de réussite, l'en-tête final serait : `X-Neptune-Status: 200 OK`. En cas d'échec, le code de réponse est un code d'erreur du moteur Neptune, tel que `X-Neptune-Status: 500 TimeLimitExceededException`.

- `X-Neptune-Detail` : est vide pour les demandes qui ont abouti. En cas d'erreur, il contient le message d'erreur JSON. Étant donné que seuls les caractères ASCII sont autorisés dans les valeurs d'en-tête HTTP, la chaîne JSON est encodée en URL. Le message d'erreur est également toujours ajouté au corps du message de réponse.

Types de médias RDF utilisés par SPARQL dans Neptune

Les données RDF (Resource Description Framework) peuvent être sérialisées de nombreuses façons différentes, la plupart pouvant être consommées ou générées par SPARQL :

Formats de sérialisation RDF utilisés par Neptune SPARQL

- RDF/XML— Sérialisation XML de RDF, définie dans la syntaxe XML RDF 1.1. Type de support : `application/rdf+xml`. Extension de fichier type : `.rdf`.
- N-Triples— Format de texte brut basé sur des lignes pour coder un graphe RDF, défini dans RDF 1.1. N-Triples Type de support : `application/n-triples`, `text/turtle` ou `text/plain`. Extension de fichier type : `.nt`.
- N-Quads— Format de texte brut basé sur des lignes pour coder un graphe RDF, défini dans RDF 1.1. N-Quads Il s'agit d'une extension de N-Triples. Type de support :`application/n-quads`, ou `text/x-nquads` lorsqu'il est codé en 7 bits US-ASCII. Extension de fichier type : `.nq`.
- Turtle : syntaxe textuelle pour RDF définie dans Turtle pour RDF 1.1 qui permet à un graphe RDF d'être complètement écrit dans un format de texte naturel et compact, avec des abréviations pour les types de données et modèles d'utilisation courants. Turtle fournit des niveaux de compatibilité avec le N-Triples format ainsi qu'avec la syntaxe à trois modèles de SPARQL. Type de support : `text/turtle`Extension de fichier type : `.ttl`.
- TriG : syntaxe textuelle pour RDF définie dans TriG pour RDF 1.1 qui permet à un graphe RDF d'être complètement écrit dans un format de texte naturel et compact, avec des abréviations pour les types de données et modèles d'utilisation courants. TriG est une extension du format Turtle. Type de support : `application/trig`. Extension de fichier type : `.trig`.
- N3 (Notation3) : langage de logique et assertion défini dans Notation3 (N3) : une syntaxe RDF lisible. N3 étend le modèle de données RDF en ajoutant des formules (des littéraux qui sont eux-mêmes des graphes), des variables, des implications logiques et des prédicats fonctionnels, et fournit une alternative syntaxique textuelle à. RDF/XML Type de support : `text/n3`. Extension de fichier type : `.n3`.
- JSON-LD— Un format de sérialisation des données et de messagerie défini dans la JSON-LD version 1.0. Type de média :. `application/ld+json` Extension de fichier type : `.jsonld`.

- TriX : sérialisation de RDF en XML, définie dans TriX : triplets RDF en XML. Type de support : `application/trix`. Extension de fichier type : `.trix`.
- Résultats JSON SPARQL : sérialisation de RDF avec le format JSON des résultats de requêtes SPARQL 1.1. Type de support : `application/sparql-results+json`. Extension de fichier type : `.srj`.
- Format binaire RDF4J : format binaire pour l'encodage des données RDF, documenté dans Format RDF binaire RDF4J. Type de support : `application/x-binary-rdf`.

Formats de sérialisation des résultats SPARQL utilisés par Neptune SPARQL

- Résultats XML SPARQL : format XML pour la liaison de variables et les formats de résultats booléens fournis par le langage de requête SPARQL, défini dans Format XML des résultats de requêtes SPARQL (deuxième édition). Type de support : `application/sparql-results+xml`. Extension de fichier type : `.srx`.
- Résultats CSV et TSV SPARQL : utilisation de valeurs séparées par des virgules ou par des tabulations pour exprimer les résultats de requêtes SPARQL des requêtes `SELECT`, définie dans Formats CSV et TSV des résultats de requêtes SPARQL 1.1. Type de support : `text/csv` pour les valeurs séparées par des virgules et `text/tab-separated-values` pour les valeurs séparées par des tabulations. Extensions de fichiers types : `.csv` pour les valeurs séparées par des virgules et `.tsv` pour les valeurs séparées par des tabulations.
- Tableau des résultats binaires : format binaire pour l'encodage de la sortie des requêtes SPARQL. Type de support : `application/x-binary-rdf-results-table`.
- Résultats JSON SPARQL : sérialisation de RDF avec le format JSON des résultats de requêtes SPARQL 1.1. Type de support : `application/sparql-results+json`.

Media-Types que Neptune peut utiliser pour importer des données RDF

Media-types soutenu par le Chargeur en vrac Neptune

- N-Triples
- N-Quads
- RDF/XML
- Turtle

Media-types que SPARQL UPDATE LOAD peut importer

- N-Triples
- N-Quads
- RDF/XML
- Turtle
- TriG
- N3
- JSON-LD

Media-Types que Neptune peut utiliser pour exporter les résultats des requêtes

Pour spécifier le format de sortie pour une réponse de requête SPARQL, envoyez un en-tête "`Accept: media-type`" avec la demande de requête. Par exemple :

```
curl -H "Accept: application/nquads" ...
```

Types de supports RDF que SPARQL SELECT peut générer à partir de Neptune

- Résultats JSON SPARQL (Ceci est la valeur par défaut)
- Résultats XML SPARQL
- Tableau de résultats binaires (type de support : `application/x-binary-rdf-results-table`)
- Comma-Separated Valeurs (CSV)
- Tab-Separated Valeurs (TSV)

Types de supports RDF que SPARQL ASK peut générer à partir de Neptune

- Résultats JSON SPARQL (Ceci est la valeur par défaut)
- Résultats XML SPARQL
- Valeur booléenne (type de support : `text/boolean`, ce qui signifie « vrai » ou « faux »)

Types de supports RDF que SPARQL CONSTRUCT peut générer à partir de Neptune

- N-Quads(Il s'agit de la valeur par défaut)

- RDF/XML
- JSON-LD
- N-Triples
- Turtle
- N3
- TriX
- TriG
- Résultats JSON SPARQL
- Format RDF binaire RDF4J

Types de supports RDF que SPARQL DESCRIBE peut générer à partir de Neptune

- N-Quads(Il s'agit de la valeur par défaut)
- RDF/XML
- JSON-LD
- N-Triples
- Turtle
- N3
- TriX
- TriG
- Résultats JSON SPARQL
- Format RDF binaire RDF4J

Utilisation de SPARQL UPDATE LOAD pour importer des données dans Neptune

La syntaxe de la commande SPARQL UPDATE LOAD est spécifiée dans la recommandation de mise à jour SPARQL 1.1 :

```
LOAD SILENT (URL of data to be loaded) INTO GRAPH (named graph into which to load the
 data)
```

- **`SILENT`** : (facultatif) fait en sorte que l'opération renvoie une réussite même en cas d'erreur lors du traitement.

Cela peut être utile lorsqu'une seule transaction contient plusieurs déclarations, comme `"LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;"`, et si vous souhaitez qu'elle soit exécutée même si certaines données distantes n'ont pas pu être traitées.

- *`URL of data to be loaded`*— (Obligatoire) Spécifie un fichier de données distant contenant les données à charger dans un graphique.

  Le fichier distant doit avoir l'une des extensions suivantes :

  - `.nt` pour NTriples.
  - `.nq` pour NQuads.
  - `.trig` pour Trig.
  - `.rdf`pour RDF/XML.
  - `.ttl` pour Turtle.
  - `.n3` pour N3.
  - `.jsonld`pour JSON-LD.
- **`INTO GRAPH`***`(named graph into which to load the data)`*— (Facultatif) Spécifie le graphique dans lequel les données doivent être chargées.

  Neptune associe chaque triplet à un graphe nommé. Vous pouvez spécifier le graphe nommé par défaut à l'aide de l'URI de remplacement, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`, comme suit :

  ```
  INTO GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
  ```

> **Note**
>
> Lorsque vous devez charger un grand nombre de données, nous vous recommandons d'utiliser le chargeur en bloc Neptune au lieu d'UPDATE LOAD. Pour plus d'informations sur le chargeur en bloc, consultez Utilisation du chargeur en vrac Amazon Neptune pour ingérer des données.

Vous pouvez utiliser `SPARQL UPDATE LOAD` pour charger des données directement à partir d'Amazon S3 ou de fichiers obtenus à partir d'un serveur web auto-hébergé. Les ressources à charger doivent se trouver dans la même région que le serveur Neptune, et le point de terminaison

pour les ressources doit être ajouté à la liste blanche dans le VPC. Pour en savoir plus sur la création d'un point de terminaison Amazon S3, consultez Création d'un point de terminaison de VPC Amazon S3.

Tous les URI `SPARQL UPDATE LOAD` doivent commencer par `https://`. Cela inclut les URL Amazon S3.

Contrairement au chargeur en bloc Neptune, un appel à `SPARQL UPDATE LOAD` est entièrement transactionnel.

Chargement des fichiers directement depuis Amazon S3 dans Neptune avec SPARQL UPDATE LOAD

Étant donné que Neptune ne vous permet pas de transmettre un rôle IAM à Amazon S3 lors de l'utilisation de SPARQL UPDATE LOAD, le compartiment Amazon S3 en question doit être public ou vous devez utiliser une URL Amazon S3 présignée dans la requête LOAD.

Pour générer une URL pré-signée pour un fichier Amazon S3, vous pouvez utiliser une AWS CLI commande comme celle-ci :

```
aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
 data to load)
```

Vous pouvez ensuite utiliser l'URL pré-signée qui en résulte dans la commande `LOAD` :

```
curl https://(a Neptune endpoint URL):8182/sparql \
  --data-urlencode 'update=load (pre-signed URL of the remote Amazon S3 file of data to
 be loaded) \
                           into graph (named graph)'
```

Pour plus d'informations, consultez Authentification des demandes : utilisation des paramètres de requête. La documentation Boto3 montre comment utiliser un script Python pour générer une URL présignée.

De plus, le type de contenu des fichiers à charger doit être défini correctement.

1. Définissez le type de contenu des fichiers lorsque vous les chargez dans Amazon S3 à l'aide du paramètre `-metadata`, comme suit :

   ```
   aws s3 cp test.nt s3://bucket-name/my-plain-text-input/test.nt --metadata Content-
   Type=text/plain
   ```

```
aws s3 cp test.rdf s3://bucket-name/my-rdf-input/test.rdf --metadata Content-
Type=application/rdf+xml
```

2. Vérifiez que les informations relatives au type de support sont réellement présentes. Exécuter :

   ```
   curl -v bucket-name/folder-name
   ```

   La sortie de cette commande doit indiquer les informations relatives au type de support que vous définissez lors du chargement des fichiers.

3. Ensuite, vous pouvez utiliser la commande `SPARQL UPDATE LOAD` pour importer ces fichiers dans Neptune :

   ```
   curl https://your-neptune-endpoint:port/sparql \
     -d "update=LOAD <https://s3.amazonaws.com/bucket-name/my-rdf-input/test.rdf>"
   ```

Les étapes ci-dessus ne fonctionnent que pour un compartiment Amazon S3 public ou pour un compartiment auquel vous accédez à l'aide d'une URL Amazon S3 présignée dans la requête LOAD.

Vous pouvez également configurer un serveur proxy web pour le charger à partir d'un compartiment Amazon S3 privé, comme indiqué ci-dessous :

Utilisation d'un serveur web pour charger des fichiers dans Neptune avec SPARQL UPDATE LOAD

1. Installez un serveur web sur un ordinateur exécuté au sein du VPC qui héberge Neptune et les fichiers à charger. Par exemple, à l'aide d'Amazon Linux, vous pouvez installer Apache comme suit :

   ```
   sudo yum install httpd mod_ssl
   sudo /usr/sbin/apachectl start
   ```

2. Définissez le(s) type(s) MIME du contenu de fichiers RDF que vous allez charger. SPARQL utilise l'en-tête `Content-type` envoyé par le serveur web pour déterminer le format d'entrée du contenu et, par conséquent, vous devez définir les types MIME correspondants pour le serveur web.

   Par exemple, supposons que vous utilisiez les extensions de fichiers suivantes pour identifier les formats de fichiers :

   - `.nt` pour NTriples.

- `.nq` pour NQuads.
- `.trig` pour Trig.
- `.rdf`pour RDF/XML.
- `.ttl` pour Turtle.
- `.n3` pour N3.
- `.jsonld`pour JSON-LD.

Si vous utilisez Apache 2 en tant que serveur web, vous devez modifier le fichier `/etc/mime.types` et ajouter les types suivants :

```
text/plain nt
application/n-quads nq
application/trig trig
application/rdf+xml rdf
application/x-turtle ttl
text/rdf+n3 n3
application/ld+json jsonld
```

3. Vérifiez que le MIME-type mappage fonctionne. Une fois que votre serveur web est opérationnel avec l'hébergement des fichiers RDF dans le(s) format(s) de votre choix, vous pouvez tester la configuration en envoyant une demande au serveur web à partir de votre hôte local.

   Par exemple, vous pouvez envoyer une requête comme celle-ci :

   ```
   curl -v http://localhost:80/test.rdf
   ```

   Ensuite, dans la sortie détaillée à partir de `curl`, vous devriez voir une ligne telle que :

   ```
   Content-Type: application/rdf+xml
   ```

   Ceci indique que le mappage du type de contenu a été défini avec succès.

4. Vous êtes maintenant prêt à charger les données à l'aide de la commande SPARQL UPDATE :

   ```
   curl https://your-neptune-endpoint:port/sparql \
       -d "update=LOAD <http://web_server_private_ip:80/test.rdf>"
   ```

> **Note**
>
> L'utilisation de `SPARQL UPDATE LOAD` peut déclencher un délai d'expiration sur le serveur web lorsque le fichier source en cours de chargement est volumineux. Neptune traite les données de fichier au fur et à mesure de leur diffusion et, pour un fichier volumineux, cela peut prendre plus de temps que le délai d'expiration configuré sur le serveur. Cela peut provoquer la fermeture de la connexion par le serveur, ce qui peut entraîner le message d'erreur suivant lorsque Neptune rencontre une fin de fichier (EOF) inattendue dans le flux :
>
> ```
> {
>   "detailedMessage":"Invalid syntax in the specified file",
>   "code":"InvalidParameterException"
> }
> ```
>
> Si vous recevez ce message et que vous ne pensez pas que votre fichier source contienne une syntaxe non valide, essayez d'augmenter les paramètres de délai d'expiration sur le serveur web. Vous pouvez également diagnostiquer le problème en activant les journaux de débogage sur le serveur et en recherchant les délais d'expiration.

Utilisation de SPARQL UPDATE UNLOAD pour supprimer des données de Neptune

Neptune fournit également une opération SPARQL personnalisée, `UNLOAD`, pour supprimer les données spécifiées dans une source distante. `UNLOAD` peut être considéré comme une contrepartie à l'opération `LOAD`. Sa syntaxe est la suivante :

```
UNLOAD SILENT (URL of the remote data to be unloaded) FROM GRAPH (named graph from
 which to remove the data)
```

- **`SILENT`** : (facultatif) fait en sorte que l'opération renvoie une réussite même en cas d'erreur lors du traitement des données.

  Cela peut être utile lorsqu'une seule transaction contient plusieurs déclarations, comme `"LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;"`, et si vous souhaitez qu'elle soit exécutée même si certaines données distantes n'ont pas pu être traitées.
- *`URL of the remote data to be unloaded`*— (Obligatoire) Spécifie un fichier de données distant contenant les données à décharger d'un graphique.

Le fichier distant doit avoir l'une des extensions suivantes (ce sont les mêmes formats que ceux pris UPDATE-LOAD en charge) :

- `.nt` pour NTriples.
- `.nq` pour NQuads.
- `.trig` pour Trig.
- `.rdf`pour RDF/XML.
- `.ttl` pour Turtle.
- `.n3` pour N3.
- `.jsonld`pour JSON-LD.

Toutes les données contenues dans ce fichier seront supprimées de votre cluster de bases de données par l'opération `UNLOAD`.

Toute authentification Amazon S3 doit être incluse dans l'URL pour que les données soient déchargées. Vous pouvez présigner un fichier Amazon S3, puis utiliser l'URL qui en résulte pour y accéder en toute sécurité. Par exemple :

```
aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
 data to unload)
```

Ensuite :

```
curl https://(a Neptune endpoint URL):8182/sparql \
  --data-urlencode 'update=unload (pre-signed URL of the remote Amazon S3 data to be
 unloaded) \
                           from graph (named graph)'
```

Pour plus d'informations, consultez Authentification des demandes : utilisation des paramètres de requête.

- **`FROM GRAPH`** *(named graph from which to remove the data)*— (Facultatif) Spécifie le graphe nommé à partir duquel les données distantes doivent être déchargées.

  Neptune associe chaque triplet à un graphe nommé. Vous pouvez spécifier le graphe nommé par défaut à l'aide de l'URI de remplacement, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`, comme suit :

```
FROM GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
```

De la même manière que LOAD correspond à INSERT DATA { *(inline data)* }, UNLOAD correspond à DELETE DATA { *(inline data)* }. Comme DELETE DATA, UNLOAD ne fonctionne pas sur les données contenant des nœuds vides.

Par exemple, si un serveur web local gère un fichier nommé `data.nt` contenant les deux triplets suivants :

```
<http://example.org/resource#a> <http://example.org/resource#p> <http://example.org/
resource#b> .
<http://example.org/resource#a> <http://example.org/resource#p> <http://example.org/
resource#c> .
```

La commande UNLOAD suivante supprime ces deux triplets du graphe nommé, `<http://example.org/graph1>` :

```
UNLOAD <http://localhost:80/data.nt> FROM GRAPH <http://example.org/graph1>
```

L'effet serait le même que si vous utilisiez la commande DELETE DATA suivante :

```
DELETE DATA {
  GRAPH <http://example.org/graph1> {
    <http://example.org/resource#a> <http://example.org/resource#p> <http://
example.org/resource#b> .
    <http://example.org/resource#a> <http://example.org/resource#p> <http://
example.org/resource#c> .
  }
}
```

Exceptions déclenchées par la commande **UNLOAD**

- **InvalidParameterException** : présence de nœuds vides dans les données. État HTTP : demande 400 incorrecte.

  Message : `Blank nodes are not allowed for UNLOAD`

- **`InvalidParameterException`** : syntaxe des données incorrecte. État HTTP : demande 400 incorrecte.

  Message : `Invalid syntax in the specified file.`

- **`UnloadUrlAccessDeniedException`** : accès refusé. État HTTP : demande 400 incorrecte.

  Message : `Update failure: Endpoint (Neptune endpoint) reported access denied error. Please verify access.`

- **`BadRequestException`** : les données distantes ne peuvent pas être récupérées. État HTTP : demande 400 incorrecte.

  Message : (dépend de la réponse HTTP).

Indicateurs de requête SPARQL

Vous pouvez utiliser des indicateurs de requête afin de spécifier des stratégies d'optimisation et d'évaluation pour une requête SPARQL particulière dans Amazon Neptune.

Les indicateurs de requête sont exprimés à l'aide de modèles de triplet supplémentaires qui sont intégrés à la requête SPARQL avec les éléments suivants :

```
scope hint value
```

- portée : détermine la partie de la requête à laquelle l'indicateur de requête s'applique, comme un certain groupe dans la requête ou la requête complète.
- hint : identifie le type d'indicateur à appliquer.
- value : détermine le comportement de l'aspect du système pris en compte.

Les indicateurs et portées de requête sont exposés sous la forme de termes prédéfinis dans l'espace de noms Amazon Neptune `http://aws.amazon.com/neptune/vocab/v01/QueryHints#`. Les exemples de cette section décrivent l'espace de noms sous la forme d'un préfixe `hint` qui est défini et inclus dans la requête :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

Par exemple, le code suivant montre comment inclure un indicateur `joinOrder` dans une requête `SELECT` :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ... {
 hint:Query hint:joinOrder "Ordered" .
 ...
}
```

La requête précédente demande au moteur Neptune d'évaluer les jointures dans la requête dans l'ordre donné et de désactiver toute réorganisation automatique.

Tenez compte des éléments suivants lorsque vous utilisez des indicateurs de requête :

- Vous pouvez combiner différents indicateurs de requête dans une seule requête. Par exemple, vous pouvez utiliser l'indicateur de requête `bottomUp` afin d'annoter une sous-requête pour une évaluation ascendante et un indicateur de requête `joinOrder` pour corriger l'ordre des jointures à l'intérieur de la sous-requête.
- Vous pouvez utiliser le même indicateur de requête plusieurs fois, dans différentes portées qui ne se chevauchent pas.
- Les indicateurs de requête sont des suggestions. Même si le moteur de requête vise généralement à prendre en compte des indicateurs de requête donnés, il peut également les ignorer.
- Les indicateurs de requête préservent la sémantique. L'ajout d'un indicateur de requête ne modifie pas la sortie de la requête (sauf pour l'ordre des résultats potentiels quand aucune garantie d'ordre n'est fournie, en d'autres termes, lorsque l'ordre des résultats n'est pas appliqué à l'aide d'ORDER BY).

Les sections suivantes fournissent plus d'informations sur les indicateurs de requête disponibles et leur utilisation dans Neptune.

Rubriques

- Portée des indicateurs de requêtes SPARQL dans Neptune
- L'indice de requête JoinOrder SPARQL
- L'indice de requête SPARQL d'EvaluationStrategy

- L'indice de requête QueryTimeout SPARQL
- L'indice de requête RangeSafe SPARQL
- L'indice de requête QueryID SPARQL
- L'indice de requête SPARQL d'UseFe
- Indicateurs de requête SPARQL utilisés avec DESCRIBE

Portée des indicateurs de requêtes SPARQL dans Neptune

Le tableau suivant montre les portées disponibles, les indicateurs associés et des descriptions pour les indicateurs de requête SPARQL dans Amazon Neptune. Le préfixe `hint` de ces entrées représente l'espace de noms Neptune des indicateurs :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

Scope	Indicateurs pris en charge	Description
`hint:Query`	joinOrder	L'indicateur de requête s'applique à toute la requête.
`hint:Query`	queryTimeout	La valeur de délai d'attente s'applique à l'ensemble de la requête.
`hint:Query`	rangeSafe	La promotion de type est désactivée pour l'ensemble de la requête.
`hint:Query`	queryId	La valeur d'ID de requête s'applique à l'ensemble de la requête.
`hint:Query`	useDFE	L'utilisation du DFE est activée (ou désactivée) pour l'ensembl e de la requête.
`hint:Group`	joinOrder	L'indicateur de requête s'applique aux éléments

Scope	Indicateurs pris en charge	Description
		de niveau supérieur du groupe spécifié, mais pas aux éléments imbriqués (comme les sous-requêtes) ou aux éléments parents.
`hint:SubQuery`	evaluationStrategy	L'indicateur est spécifié et appliqué à une sous-requ ête SELECT imbriquée. La sous-requête est évaluée de façon indépendante, sans tenir compte des solutions calculées avant la sous-requ ête.

L'indice de **`requête JoinOrder`** SPARQL

Lorsque vous soumettez une requête SPARQL, le moteur de requête Amazon Neptune étudie sa structure. Il réorganise les parties de la requête et tente de réduire au maximum la quantité de travail nécessaire pour l'évaluation et le temps de réponse de la requête.

Par exemple, une séquence de modèles de triplet connectés n'est généralement pas évaluée dans l'ordre donné. Elle est réorganisée à l'aide d'heuristique et de statistiques telles que la sélectivité des différents modèles et la façon dont ils sont connectés via des variables partagées. En outre, si votre requête contient des modèles plus complexes comme des sous-requêtes, des filtres FILTER, ou encore des blocs OPTIONAL ou MINUS complexes, le moteur de requête Neptune les réorganise dans la mesure du possible avec pour but un ordre d'évaluation efficace.

Pour les requêtes plus complexes, l'ordre dans lequel Neptune choisit d'évaluer la requête peut ne pas toujours être optimal. Par exemple, Neptune peut manquer des caractéristiques spécifiques aux données d'instance (par exemple, l'exécution de nœuds Power dans le graphe) qui émergent au cours de l'évaluation de la requête.

Si vous connaissez les caractéristiques exactes des données et que vous souhaitez imposer manuellement l'ordre de l'exécution de la requête, utilisez l'indicateur de requête Neptune `joinOrder` pour demander que la requête soit évaluée dans l'ordre donné.

Syntaxe des `indices JoinOrder` SPARQL

L'indicateur de requête `joinOrder` est spécifié en tant que modèle de triplet inclus dans une requête SPARQL.

Pour plus de clarté, la syntaxe suivante utilise un préfixe `hint` défini et inclus dans la requête pour spécifier l'espace de noms d'indicateur de requête Neptune :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
scope hint:joinOrder "Ordered" .
```

Portées disponibles

- `hint:Query`
- `hint:Group`

Pour plus d'informations sur les portées d'indicateur de requête, consultez Portée des indicateurs de requêtes SPARQL dans Neptune.

Exemple d'`astuce JoinOrder` SPARQL

Cette section montre une requête écrite avec et sans l'indicateur de requête `joinOrder`, ainsi que les optimisations associées.

Pour cet exemple, supposons que l'ensemble de données contient les éléments suivants :

- Une personne unique nommée `John` qui `:likes` 1 000 personnes, y compris `Jane`.
- Une personne unique nommée `Jane` qui `:likes` 10 personnes, y compris `John`.

Aucun indicateur de requête

La requête SPARQL suivante extrait toutes les paires de personnes nommées `John` et `Jane` qui sont amies à partir d'un ensemble de données de réseau social :

```
PREFIX : <https://example.com/>
SELECT ?john ?jane {
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
```

```
}
```

Le moteur de requête Neptune peut évaluer les déclarations dans un ordre différent de celui qui est écrit. Par exemple, il peut choisir d'évaluer dans l'ordre suivant :

1. Trouver toutes les personnes nommées `John`.
2. Trouver toutes les personnes connectées à `John` par une arête `:likes`.
3. Filtrer cet ensemble par les personnes nommées `Jane`.
4. Filtrer cet ensemble par les personnes connectées à `John` par une arête `:likes`.

Selon l'ensemble de données, l'évaluation dans cette commande se traduit par 1 000 entités extraites lors de la deuxième étape. La troisième étape affine l'extraction à un nœud unique, `Jane`. L'étape finale détermine ensuite que `Jane` `:likes` également le nœud `John`.

Indicateur de requête

Il serait préférable de démarrer avec le nœud `Jane`, car Jane n'a que 10 arêtes `:likes` sortantes. Cela réduit la quantité de travail pendant l'évaluation de la requête en évitant l'extraction des 1 000 entités au cours de la deuxième étape.

L'exemple suivant utilise l'indicateur de requête joinOrder afin de s'assurer que le nœud `Jane` et ses arêtes sortantes sont traités en premier en désactivant toute la réorganisation automatique des jointures pour la requête :

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?john ?jane {
  hint:Query hint:joinOrder "Ordered" .
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Un scénario réel applicable peut être une application de réseau social où les utilisateurs du réseau sont classés soit comme des influenceurs avec de nombreuses connexions, soit comme des utilisateurs normaux avec peu de connexions. Dans un tel scénario, vous pouvez vous assurer que l'utilisateur normal (`Jane`) est traité avant l'influenceur (`John`) dans une requête similaire à l'exemple précédent.

Indicateur de requête et réorganisation

Vous pouvez encore améliorer cet exemple. Si vous savez que l'attribut `:name` est unique à un seul nœud, vous pouvez accélérer la requête en réorganisant et en utilisant l'indicateur de requête `joinOrder`. Cette étape permet de s'assurer que les nœuds uniques sont extraits en premier.

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?john ?jane {
  hint:Query hint:joinOrder "Ordered" .
  ?person1 :name "Jane" .
  ?person2 :name "John" .
  ?person1 :likes ?person2 .
  ?person2 :likes ?person1 .
}
```

Dans ce cas, vous pouvez réduire la requête suivante aux actions uniques suivantes dans chaque étape :

1. Rechercher le seul nœud de personne avec l'attribut `:name Jane`.
2. Rechercher le seul nœud de personne avec l'attribut `:name John`.
3. Vérifier que le premier nœud est connecté au deuxième nœud avec une arête `:likes`.
4. Vérifier que le deuxième nœud est connecté au premier nœud avec une arête `:likes`.

> **Important**
>
> Si vous ne choisissez pas l'ordre approprié, l'indicateur de requête `joinOrder` peut entraîner des baisses significatives de performance. Par exemple, l'exemple précédent serait inefficace si les attributs `:name` n'étaient pas uniques. Si les 100 nœuds étaient nommés `Jane` et si tous les 1 000 nœuds étaient nommés `John`, la requête auraient à vérifier 1 000 * 100 (100 000) paires pour les arêtes `:likes`.

L'indice de requête **`SPARQL d'EvaluationStrategy`**

L'indicateur de requête `evaluationStrategy` indique au moteur de requête Amazon Neptune que le fragment de la requête annotée doit être évalué de bas en haut en tant qu'unité indépendante.

Cela signifie qu'aucune des solutions des étapes d'évaluation précédentes n'est utilisée pour calculer le fragment de requête. Le fragment de requête est évalué en tant qu'unité autonome, et ses solutions produites sont jointes au reste de la requête une fois que celle-ci est calculée.

L'utilisation de l'indicateur de requête `evaluationStrategy` implique un plan de requête bloquant (non en pipeline), ce qui signifie que les solutions du fragment annoté avec l'indicateur de requête sont matérialisées et mises en tampon dans la mémoire principale. L'utilisation de l'indicateur de requête peut se traduire par une augmentation importante de la quantité de mémoire principale requise pour évaluer la requête, surtout si le fragment de requête annoté calcule un grand nombre de résultats.

Syntaxe des indices **`SPARQL d'EvaluationStrategy`**

L'indicateur de requête `evaluationStrategy` est spécifié en tant que modèle de triplet inclus dans une requête SPARQL.

Pour plus de clarté, la syntaxe suivante utilise un préfixe `hint` défini et inclus dans la requête pour spécifier l'espace de noms d'indicateur de requête Neptune :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
hint:SubQuery hint:evaluationStrategy "BottomUp" .
```

Portées disponibles

- `hint:SubQuery`

> **Note**
>
> Cet indicateur de requête est pris en charge uniquement dans les sous-requêtes imbriquées.

Pour plus d'informations sur les portées d'indicateur de requête, consultez Portée des indicateurs de requêtes SPARQL dans Neptune.

Exemple d'astuce **`SPARQL pour EvaluationStrategy`**

Cette section montre une requête écrite avec et sans l'indicateur de requête `evaluationStrategy`, ainsi que les optimisations associées.

Pour cet exemple, supposons que l'ensemble de données a les caractéristiques suivantes :

- Il contient de 1 000 arêtes étiquetées `:connectedTo`.
- Chaque nœud `component` est connecté en moyenne à 100 autres nœuds `component`.
- Le nombre typique de connexions cycliques à quatre tronçons entre les nœuds est de 100 environ.

Comme exemple typique, l'indicateur `evaluationStrategy` peut s'avérer utile pour optimiser des modèles de requêtes qui contiennent des cycles.

Aucun indicateur de requête

La requête SPARQL suivante extrait tous les nœuds `component` qui sont connectés cycliquement les uns aux autres via quatre tronçons :

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
  ?component1 :connectedTo ?component2 .
  ?component2 :connectedTo ?component3 .
  ?component3 :connectedTo ?component4 .
  ?component4 :connectedTo ?component1 .
}
```

L'approche du moteur de requête Neptune consiste à évaluer cette requête à l'aide des étapes suivantes :

- Extraire la totalité des 1 000 arêtes `connectedTo` dans le graphe.
- Multiplier par 100 (le nombre d'arêtes `connectedTo` sortantes de component2).

  Résultats intermédiaires : 100 000 nœuds.
- Multiplier par 100 (le nombre d'arêtes `connectedTo` sortantes de component3).

  Résultats intermédiaires : 10 000 000 nœuds.
- Analyser les 10 000 000 nœuds pour la fermeture du cycle.

Cela se traduit par un plan de requête de streaming ayan une quantité constante de mémoire principale.

Indicateur de requête et sous-requêtes

Il se peut que vous souhaitiez faire une compromis sur l'espace mémoire principale pour accélérer le calcul. En réécrivant la requête à l'aide d'un indicateur de requête `evaluationStrategy`, vous pouvez forcer le moteur à calculer une jointure entre deux sous-ensembles matérialisés plus petits.

```
PREFIX : <https://example.com/>
          PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
  {
    SELECT * WHERE {
      hint:SubQuery hint:evaluationStrategy "BottomUp" .
      ?component1 :connectedTo ?component2 .
      ?component2 :connectedTo ?component3 .
    }
  }
  {
    SELECT * WHERE {
      hint:SubQuery hint:evaluationStrategy "BottomUp" .
      ?component3 :connectedTo ?component4 .
      ?component4 :connectedTo ?component1 .
    }
  }
}
```

Au lieu d'évaluer les modèles de triplet dans l'ordre tout en utilisant de manière itérative les résultats des modèles de triplet précédents comme entrée pour les modèles suivantes, avec l'indicateur `evaluationStrategy`, les deux sous-requêtes sont évaluées de manière indépendante. Les deux sous-requêtes produisent 100 000 nœuds pour les résultats intermédiaires, qui sont ensuite joints pour former la sortie finale.

En particulier, lorsque vous exécutez Neptune sur des types d'instances de plus grande taille, stocker temporairement ces deux sous-ensembles de 100 000 nœuds en mémoire principale augmente l'utilisation de la mémoire tout en accélérant l'évaluation de manière significative.

L'indice de **`requête QueryTimeout SPARQL`**

L'indicateur de requête `queryTimeout` spécifie un délai d'expiration qui est inférieur à la valeur `neptune_query_timeout` définie dans le groupe de paramètres de base de données.

Si la requête se termine en raison de cet indicateur, une exception `TimeLimitExceededException` est déclenchée avec le message `Operation terminated (deadline exceeded)`. L'opportunité de réessayer une requête expirée dépend de la nature de

l'échec et de votre charge de travail. Pour de plus amples informations, consultez the section called "Exceptions et nouvelles tentatives".

Syntaxe des **`indices QueryTimeout SPARQL`**

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ... WHERE {
    hint:Query hint:queryTimeout 10 .
    # OR
    hint:Query hint:queryTimeout "10" .
    # OR
    hint:Query hint:queryTimeout "10"^^xsd:integer .
 ...
}
```

La valeur de délai d'attente est exprimée en millisecondes.

La valeur de délai d'attente doit être inférieure à la valeur `neptune_query_timeout` définie dans le groupe de paramètres de base de données. Sinon, une exception `MalformedQueryException` est déclenchée avec le message `Malformed query: Query hint 'queryTimeout' must be less than neptune_query_timeout DB Parameter Group`.

L'indicateur de requête `queryTimeout` doit être spécifié dans la clause `WHERE` de la requête principale ou dans la clause `WHERE` de l'une des sous-requêtes, comme dans l'exemple ci-dessous.

Il ne doit être défini qu'une seule fois dans toutes les sections queries/subqueries et mises à jour de SPARQL (telles que INSERT et DELETE). Sinon, une exception `MalformedQueryException` est déclenchée avec le message `Malformed query: Query hint 'queryTimeout' must be set only once`.

Portées disponibles

L'indicateur `queryTimeout` peut être appliqué aux requêtes et mises à jour SPARQL.

- Dans une requête SPARQL, il peut figurer dans la clause WHERE de la requête principale ou d'une sous-requête.
- Dans une mise à jour SPARQL, il peut être défini dans la clause INSERT, DELETE ou WHERE. S'il existe plusieurs clauses de mise à jour, il ne peut être défini que dans l'une d'elles.

Pour plus d'informations sur les portées d'indicateur de requête, consultez Portée des indicateurs de requêtes SPARQL dans Neptune.

Exemple d'**indice QueryTimeout SPARQL**

Voici un exemple d'utilisation de `hint:queryTimeout` dans la clause `WHERE` principale d'une requête `UPDATE` :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
INSERT {
    ?s ?p ?o
} WHERE {
    hint:Query hint:queryTimeout 100 .
    ?s ?p ?o .
}
```

Ici, le `hint:queryTimeout` est dans la clause `WHERE` d'une sous-requête :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
   ?s ?p ?o .
   {
      SELECT ?s WHERE {
         hint:Query hint:queryTimeout 100 .
         ?s ?p1 ?o1 .
      }
   }
}
```

L'indice de **requête RangeSafe** SPARQL

Utilisez cet indicateur de requête pour désactiver la promotion de type pour une requête SPARQL.

Lorsque vous soumettez une requête SPARQL qui filtre une plage ou une valeur numérique avec `FILTER`, le moteur de requêtes Neptune doit normalement utiliser la promotion de type lorsqu'il exécute la requête. Autrement dit, il doit examiner les valeurs de tous les types susceptibles de contenir la valeur sur laquelle porte le filtre.

Par exemple, si vous filtrez les valeurs égales à 55, le moteur doit rechercher les entiers égaux à 55, les entiers longs égaux à 55L, les nombres flottants égaux à 55,0, etc. Chaque promotion de type implique une recherche supplémentaire au niveau du stockage, ce qui peut entraîner un délai étonnamment long pour terminer une requête apparemment simple.

Souvent, la promotion de type n'est pas nécessaire, car vous savez déjà que vous n'avez besoin de trouver que les valeurs d'un type spécifique. Dans ce cas, vous pouvez accélérer considérablement les requêtes en utilisant l'indicateur de requête `rangeSafe` afin de désactiver la promotion de type.

Syntaxe des **`indices RangeSafe SPARQL`**

L'indicateur de requête `rangeSafe` utilise la valeur `true` pour désactiver la promotion de type. Il accepte également la valeur `false` (valeur par défaut).

Exemple. L'exemple suivant montre comment désactiver la promotion de type lors du filtrage d'une valeur entière o supérieure à 1 :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
   ?s ?p ?o .
   hint:Prior hint:rangeSafe 'true' .
   FILTER (?o > '1'^^<http://www.w3.org/2001/XMLSchema#int>)
```

L'indice de **`requête QueryID`** SPARQL

Utilisez cet indicateur de requête pour attribuer votre propre valeur queryId à une requête SPARQL.

Exemple :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * WHERE {
  hint:Query hint:queryId "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
  {?s ?p ?o}}
```

La valeur que vous attribuez doit être unique pour toutes les requêtes de la base de données Neptune.

L'indice de **`requête SPARQL d'UseFe`**

Utilisez cet indicateur de requête afin d'activer l'utilisation du DFE pour exécuter la requête. Par défaut, Neptune n'utilise pas le DFE sans que cet indicateur de requête ne soit défini sur `true`, car le paramètre d'instance neptune_dfe_query_engine est défini par défaut sur `viaQueryHint`. Si vous définissez ce paramètre d'instance sur `enabled`, le moteur DFE est utilisé pour toutes les requêtes, à l'exception de celles dont l'indicateur de requête `useDFE` est défini sur `false`.

Exemple d'activation de l'utilisation du DFE pour une requête :

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>

SELECT ?john ?jane
{
  hint:Query hint:useDFE true .
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Indicateurs de requête SPARQL utilisés avec DESCRIBE

Une requête SPARQL `DESCRIBE` fournit un mécanisme flexible permettant de demander des descriptions de ressources. Cependant, les spécifications SPARQL ne définissent pas la sémantique précise de `DESCRIBE`.

À partir de la version 1.2.0.2 du moteur, Neptune prend en charge plusieurs modes et algorithmes `DESCRIBE` adaptés à différentes situations.

Cet exemple de jeu de données permet d'illustrer les différents modes :

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <https://example.com/> .

:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JohnDoe :firstName "John" .
:JaneDoe :knows _:b1 .
_:b1 :knows :RichardRoe .

:RichardRoe :knows :JaneDoe .
:RichardRoe :firstName "Richard" .

_:s1 rdf:type rdf:Statement .
_:s1 rdf:subject :JaneDoe .
_:s1 rdf:predicate :knows .
_:s1 rdf:object :JohnDoe .
_:s1 :knowsFrom "Berlin" .

:ref_s2 rdf:type rdf:Statement .
```

```
:ref_s2 rdf:subject :JaneDoe .
:ref_s2 rdf:predicate :knows .
:ref_s2 rdf:object :JohnDoe .
:ref_s2 :knowsSince 1988 .
```

Les exemples ci-dessous supposent qu'une description de la ressource `:JaneDoe` est demandée à l'aide d'une requête SPARQL comme celle-ci :

```
DESCRIBE <https://example.com/JaneDoe>
```

L'indice de **`requête SPARQL DescribeMode`**

L'indicateur de requête `hint:describeMode` SPARQL est utilisé pour sélectionner l'un des modes `DESCRIBE` SPARQL suivants pris en charge par Neptune :

Mode **`ForwardOneStep`** DESCRIBE

Vous invoquez le mode `ForwardOneStep` avec l'indicateur de requête `describeMode` comme suit :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "ForwardOneStep"
}
```

Le mode `ForwardOneStep` renvoie uniquement les attributs et les liens de transfert de la ressource à décrire. Dans le cas de cet exemple, cela signifie qu'il renvoie les triplets dont la ressource à décrire, `:JaneDoe`, est le sujet :

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b301990159 .
```

Notez que la requête DESCRIBE peut renvoyer des triplets avec des nœuds vides (par exemple, `_:b301990159`), qui ont des ID différents à chaque fois par rapport au jeu de données en entrée.

Mode **`SymmetricOneStep`** DESCRIBE

`SymmetricOneStep` est le mode DESCRIBE par défaut si vous ne fournissez aucun indicateur de requête. Vous pouvez également l'invoquer explicitement avec l'indicateur de requête `describeMode` comme ceci :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SymmetricOneStep"
}
```

Sous la sémantique `SymmetricOneStep`, `DESCRIBE` renvoie les attributs, les liens de transfert et les liens inverses de la ressource à décrire :

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b318767375 .

_:b318767631 rdf:subject :JaneDoe .

:RichardRoe :knows :JaneDoe .

:ref_s2 rdf:subject :JaneDoe .
```

Le mode DESCRIPTION LIMITÉE CONCISE (**`CBD`**) DESCRIBE

Le mode `CBD` (Concise Bounded Description) est invoqué à l'aide de l'indicateur de requête `describeMode` suivant :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "CBD"
}
```

Sous la sémantique `CBD`, `DESCRIBE` renvoie la description CBD (telle que définie par le W3C) de la ressource à décrire :

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b285212943 .
_:b285212943 :knows :RichardRoe .

_:b285213199 rdf:subject :JaneDoe .
_:b285213199 rdf:type rdf:Statement .
_:b285213199 rdf:predicate :knows .
```

```
_:b285213199 rdf:object :JohnDoe .
_:b285213199 :knowsFrom "Berlin" .

:ref_s2 rdf:subject :JaneDoe .
```

La description CBD d'une ressource RDF (c'est-à-dire un nœud dans un graphe RDF) est le plus petit sous-graphe pouvant être autonome centré sur ce nœud. En pratique, cela signifie que si vous considérez ce graphe comme un arbre, avec le nœud désigné comme racine, il n'y a pas de nœuds vides comme feuilles de cet arbre. Comme les nœuds vides ne peuvent pas être traités de manière externe ni utilisés dans les requêtes ultérieures, il ne suffit pas de parcourir le graphe pour trouver le ou les prochains sauts individuels à partir du nœud actuel. Vous devez également aller assez loin pour trouver ce qui pourra être utilisé dans les requêtes ultérieures (à savoir autre chose qu'un nœud vide).

Calcul de la valeur CBD

Avec un nœud particulier (le nœud ou la racine de départ) dans le graphe RDF source, la valeur CBD de ce nœud est calculée comme suit :

1. Incluez dans le sous-graphe toutes les déclarations du graphe source dont le sujet de la déclaration est le nœud de départ.
2. De manière récursive, pour toutes les déclarations du sous-graphe contenant jusqu'à présent un objet de nœud vide, incluez dans le sous-graphe toutes les déclarations du graphe source dont le sujet de la déclaration correspond à ce nœud vide, et qui ne sont pas déjà incluses dans le sous-graphe.
3. De manière récursive, pour toutes les déclarations incluses dans le sous-graphe jusqu'à présent, pour toutes les réifications de ces déclarations dans le graphe source, incluez la valeur CBD en commençant par le nœud `rdf:Statement` de chaque réification.

Il en résulte un sous-graphe où les nœuds de l'objet sont soit des références IRI ou des littéraux, soit des nœuds vides ne faisant l'objet d'aucune déclaration dans le graphe. Notez que le CBD ne peut pas être calculé à l'aide d'une seule requête SPARQL SELECT ou CONSTRUCT.

Le mode DESCRIBE de description symétrique concise et bornée (**`SCBD`**)

Le mode `SCBD` (Symetric Concise Bounded Description) est invoqué à l'aide de l'indicateur de requête `describeMode` suivant :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

```
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SCBD"
}
```

Sous la sémantique `SCBD`, `DESCRIBE` renvoie la description SCBD (telle que définie par le W3C dans Description des jeux de données liés avec le vocabulaire VoID) :

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b335544591 .
_:b335544591 :knows :RichardRoe .

:RichardRoe :knows :JaneDoe .

_:b335544847 rdf:subject :JaneDoe .
_:b335544847 rdf:type rdf:Statement .
_:b335544847 rdf:predicate :knows .
_:b335544847 rdf:object :JohnDoe .
_:b335544847 :knowsFrom "Berlin" .

:ref_s2 rdf:subject :JaneDoe .
```

L'avantage des valeurs CBD et SCBD par rapport aux modes `ForwardOneStep` et `SymmetricOneStep` est que les nœuds vides sont toujours étendus pour inclure leur représentation. Cet avantage est de taille, car vous ne pouvez pas interroger un nœud vide à l'aide de SPARQL. En outre, les modes CBD et SCBD prennent également en compte les réifications.

Notez que l'indicateur de requête `describeMode` peut également faire partie d'une clause `WHERE` :

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE ?s
WHERE {
  hint:Query hint:describeMode "CBD" .
  ?s rdf:type <https://example.com/Person>
}
```

L'astuce de **`description`** de la requête IterationLimit SPARQL

L'indicateur de requête `hint:describeIterationLimit` SPARQL fournit une contrainte facultative sur le nombre maximum d'extensions itératives à effectuer pour les algorithmes itératifs DESCRIBE tels que CBD et SCBD.

Les limites du mode DESCRIBE doivent s'appliquer simultanément. Par conséquent, si la limite d'itération et la limite de déclarations sont spécifiées, elles doivent toutes deux être respectées avant que la requête DESCRIBE ne soit interrompue.

La valeur par défaut est 5. Vous pouvez la définir sur ZÉRO (0) pour ne pas spécifier de limite du nombre d'extensions itératives.

L'astuce de **`description`** de la requête StatementLimit SPARQL

L'indicateur de requête SPARQL `hint:describeStatementLimit` fournit une contrainte facultative sur le nombre maximum de déclarations pouvant être présentes dans une réponse à une requête DESCRIBE. Il n'est appliqué qu'aux algorithmes itératifs DESCRIBE tels que CBD et SCBD.

Les limites du mode DESCRIBE doivent s'appliquer simultanément. Par conséquent, si la limite d'itération et la limite de déclarations sont spécifiées, elles doivent toutes deux être respectées avant que la requête DESCRIBE ne soit interrompue.

La valeur par défaut est 5 000. Vous pouvez la définir sur ZÉRO (0) pour ne pas limiter le nombre de déclarations renvoyées.

Comportement de SPARQL DESCRIBE par rapport au graphe par défaut

Le type de requête SPARQL `DESCRIBE` vous permet de récupérer des informations sur les ressources sans connaître la structure des données et sans avoir à composer de requête. La manière dont ces informations sont assemblées dépend de l'implémentation SPARQL. Neptune fournit plusieurs indicateurs de requête qui invoquent différents modes et algorithmes à utiliser par `DESCRIBE`.

Dans l'implémentation de Neptune, quel que soit le mode, `DESCRIBE` utilise uniquement les données présentes dans le graphe par défaut SPARQL. Cela est cohérent avec la façon dont SPARQL traite les jeux de données (voir Spécification des jeux de données RDF dans la spécification SPARQL).

Dans Neptune, le graphe par défaut contient tous les triplets uniques issus de l'union de tous les graphes nommés de la base de données, sauf si des graphes nommés particuliers sont spécifiés à l'aide de clauses. `FROM` and/or `FROM NAMED` Toutes les données RDF de Neptune sont stockées dans un graphe nommé. Si un triplet est inséré sans contexte de graphe nommé, Neptune le stocke dans un graphe nommé désigné `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Lorsqu'un ou plusieurs graphes nommés sont spécifiés à l'aide de la clause `FROM`, le graphe par défaut est l'union de tous les triplets uniques de ces graphes nommés. S'il n'y a aucune clause `FROM` et qu'il existe une ou plusieurs clauses `FROM NAMED`, le graphe par défaut est vide.

Exemples de SPARQL **`DESCRIBE`**

Prenons les données suivantes :

```
PREFIX ex: <https://example.com/>

GRAPH ex:g1 {
    ex:s ex:p1 "a" .
    ex:s ex:p2 "c" .
}

GRAPH ex:g2 {
    ex:s ex:p3 "b" .
    ex:s ex:p2 "c" .
}

ex:s ex:p3 "d" .
```

Pour cette requête :

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM ex:g1
FROM NAMED ex:g2
WHERE {
  GRAPH ex:g2 { ?s ?p "b" . }
}
```

Neptune renvoie :

```
ex:s ex:p1 "a" .
ex:s ex:p2 "c" .
```

Ici, le modèle de graphe `GRAPH ex:g2 { ?s ?p "b" }` est d'abord évalué, ce qui entraîne des liaisons pour `?s`, puis la partie `DESCRIBE` est évaluée par rapport au graphe par défaut, qui n'est plus qu'`ex:g1`.

Toutefois, pour cette requête :

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM NAMED ex:g1
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

Neptune ne renverrait rien, car lorsqu'une clause `FROM NAMED` est présente sans aucune clause `FROM`, le graphe par défaut est vide.

Dans la requête suivante, `DESCRIBE` est utilisé en l'absence de clause `FROM` ou `FROM NAMED` :

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

Dans ce cas, le graphe par défaut est composé de tous les triplets uniques issus de l'union de tous les graphes nommés de la base de données (officiellement, la fusion RDF). Neptune renverrait donc :

```
ex:s ex:p1 "a" .
ex:s ex:p2 "c" .
ex:s ex:p3 "b" .
ex:s ex:p3 "d" .
```

API de statut des requêtes SPARQL

Pour obtenir le statut des requêtes SPARQL, envoyez une demande HTTP `GET` ou `POST` pour effectuer une requête au point de terminaison `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql/status`.

Paramètres des demandes de statut des requêtes SPARQL

queryId (facultatif)

ID d'une requête SPARQL en cours d'exécution. Affiche uniquement le statut de la requête indiquée.

Syntaxe des réponses de statut des requêtes SPARQL

```
{
```

```
    "acceptedQueryCount": integer,
    "runningQueryCount": integer,
    "queries": [
      {
        "queryId":"guid",
        "queryEvalStats":
          {
            "subqueries": integer,
            "elapsed": integer,
            "cancelled": boolean
          },
        "queryString": "string"
      }
    ]
}
```

Valeurs des réponses de statut des requêtes SPARQL

accepté QueryCount

Nombre de requêtes acceptées depuis le dernier redémarrage du moteur Neptune.

course QueryCount

Nombre de requêtes SPARQL en cours d'exécution.

queries

Liste des requêtes SPARQL actuelles.

queryId

GUID de la requête. Neptune attribue automatiquement cette valeur d'ID à chaque requête, mais vous pouvez également attribuer votre propre ID (voir Injection d'un ID personnalisé dans une requête Neptune Gremlin ou SPARQL).

requête EvalStats

Statistiques pour cette requête.

subqueries

Nombre de sous-requêtes de cette requête.

elapsed

Nombre de microsecondes d'exécution de la requête jusqu'ici.

cancelled

True indique que la requête a été annulée.

queryString

Requête soumise.

Exemple de statut des requêtes SPARQL

Voici un exemple de commande de demande du statut utilisant `curl` et la demande HTTP `GET`.

```
curl https://your-neptune-endpoint:port/sparql/status
```

Cette sortie affiche une seule requête en cours d'exécution.

```
{
    "acceptedQueryCount":9,
    "runningQueryCount":1,
    "queries": [
        {
            "queryId":"fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
            "queryEvalStats":
                {
                    "subqueries": 0,
                    "elapsed": 29256,
                    "cancelled": false
                },
            "queryString": "SELECT ?s ?p ?o WHERE {?s ?p ?o}"
        }
    ]
}
```

Annulation de requêtes SPARQL

Pour obtenir le statut des requêtes SPARQL, envoyez une demande HTTP `GET` ou `POST` pour effectuer une requête au point de terminaison `https://your-neptune-endpoint:port/sparql/status`.

Paramètres des demandes d'annulation des requêtes SPARQL

cancelQuery

(Obligatoire) Indique à la commande d'état d'annuler une requête. Ce paramètre n'est pas défini sur une valeur.

queryId

(Obligatoire) ID de la requête SPARQL en cours d'exécution à annuler.

silent

(Facultatif) Si `silent=true`, la requête en cours d'exécution est annulée et le code de réponse HTTP est 200. Si `silent` n'est pas présent ou si `silent=false`, la requête est annulée avec un code d'état HTTP 500.

Exemple d'annulation de requêtes SPARQL

Exemple 1 : Annulation avec **`silent=false`**

Voici un exemple de commande d'état utilisant `curl` pour annuler une requête avec le paramètre `silent` défini sur `false` :

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
  -d "silent=false"
```

Sauf si la requête a déjà commencé à diffuser des résultats, la requête annulée renvoie alors un code HTTP 500 avec une réponse comme celle-ci :

```
{
  "code": "CancelledByUserException",
  "requestId": "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47",
  "detailedMessage": "Operation terminated (cancelled by user)"
}
```

Si la requête a déjà renvoyé un code HTTP 200 (OK) et a commencé à diffuser des résultats avant d'être annulée, les informations d'exception de délai d'attente sont envoyées au flux de sortie normal.

Exemple 2 : Annulation avec **`silent=true`**

Voici un exemple de la même commande d'état que ci-dessus, mais avec le paramètre `silent` défini sur `true` :

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
  -d "silent=true"
```

Cette commande renvoie la même réponse que lorsque `silent=false`, mais la requête annulée renvoie désormais un code HTTP 200 avec une réponse similaire à celle-ci :

```
{
  "head" : {
    "vars" : [ "s", "p", "o" ]
  },
  "results" : {
    "bindings" : [ ]
  }
}
```

Utilisation du protocole HTTP SPARQL 1.1 Graph Store (GSP) dans Amazon Neptune

Dans la recommandation du protocole HTTP SPARQL 1.1 Graph Store, le W3C a défini un protocole HTTP pour gérer les graphes RDF. Il définit les opérations permettant de supprimer, de créer et de remplacer le contenu d'un graphe RDF ainsi que d'ajouter des instructions RDF au contenu existant.

Le protocole Graph Store (GSP) fournit un moyen pratique de manipuler l'intégralité du graphe sans avoir à écrire de requêtes SPARQL complexes.

Neptune prend pleinement en charge ce protocole.

Le point de terminaison du protocole Graph Store (GSP) est le suivant :

```
https://your-neptune-cluster:port/sparql/gsp/
```

Pour accéder au graphe par défaut avec le protocole GPS, utilisez :

```
https://your-neptune-cluster:port/sparql/gsp/?default
```

Pour accéder à un graphe nommé avec le protocole GPS, utilisez :

```
https://your-neptune-cluster:port/sparql/gsp/?graph=named-graph-URI
```

Détails particuliers de la mise en œuvre Neptune du protocole GSP

Neptune met pleinement en œuvre la recommandation du W3C qui définit le protocole GPS. Cependant, il existe quelques situations que cette spécification ne couvre pas.

L'un d'entre elles est le cas où une demande PUT ou POST spécifie dans le corps de la demande un ou plusieurs graphes nommés qui diffèrent du graphe spécifié par l'URL de la demande. Cela ne peut se produire que lorsque le format RDF du corps de la requête prend en charge les graphes nommés, par exemple en utilisant `Content-Type: application/n-quads` ou `Content-Type: application/trig`.

Dans ce cas, Neptune ajoute ou met à jour tous les graphes nommés présents dans le corps, ainsi que le graphe nommé spécifié dans l'URL.

Supposons, par exemple, qu'en partant d'une base de données vide, vous envoyiez une demande PUT pour effectuer l'upsert de votes dans trois graphes. L'un, nomméurn:votes, contient tous les votes pour toutes les années électorales. Les deux autres, nommés `urn:votes:2005` eturn:votes:2019, contiennent les votes relatifs à des années électorales spécifiques. La demande et sa charge utile sont comme suit :

```
PUT "http://your-Neptune-cluster:port/sparql/gsp/?graph=urn:votes"
  Host: example.com
  Content-Type: application/n-quads

  PAYLOAD:

  <urn:JohnDoe> <urn:votedFor> <urn:Labour> <urn:votes:2005>
  <urn:JohnDoe> <urn:votedFor> <urn:Conservative> <urn:votes:2019>
  <urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes:2005>
  <urn:JaneSmith> <urn:votedFor> <urn:Conservative> <urn:votes:2019>
```

Une fois la demande exécutée, les données de la base de données se présentent comme suit :

```
<urn:JohnDoe>   <urn:votedFor> <urn:Labour>          <urn:votes:2005>
```

```
<urn:JohnDoe>   <urn:votedFor> <urn:Conservative>     <urn:votes:2019>
<urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes:2005>
<urn:JaneSmith> <urn:votedFor> <urn:Conservative>     <urn:votes:2019>
<urn:JohnDoe>   <urn:votedFor> <urn:Labour>           <urn:votes>
<urn:JohnDoe>   <urn:votedFor> <urn:Conservative>     <urn:votes>
<urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes>
<urn:JaneSmith> <urn:votedFor> <urn:Conservative>     <urn:votes>
```

Une autre situation ambiguë est celle où plusieurs graphes sont spécifiés dans l'URL de la demande elle-même, en utilisant `PUT`, `POST`, `GET` ou `DELETE`. Par exemple :

```
POST "http://your-Neptune-cluster:port/sparql/gsp/?
graph=urn:votes:2005&graph=urn:votes:2019"
```

Ou:

```
GET "http://your-Neptune-cluster:port/sparql/gsp/?default&graph=urn:votes:2019"
```

Dans ce cas, Neptune renvoie un code HTTP 400 avec un message indiquant qu'un seul graphe peut être spécifié dans l'URL de la demande.

Analyse de l'exécution des requêtes Neptune à l'aide de SPARQL : explication

Amazon Neptune a ajouté une fonctionnalité SPARQL nommée explain. Cette fonctionnalité est un outil en libre-service qui vous aide à comprendre l'approche d'exécution adoptée par le moteur Neptune. Vous l'appelez en ajoutant un paramètre `explain` à un appel HTTP qui soumet une requête SPARQL.

La fonction `explain` fournit des informations sur la structure logique des plans d'exécution de requête. Vous pouvez utiliser ces informations pour identifier les goulots d'étranglement d'évaluation et d'exécution potentiels. Vous pouvez ensuite utiliser des indicateurs de requête pour améliorer vos plans d'exécution de requêtes.

Rubriques

- Fonctionnement du moteur de requête SPARQL dans Neptune
- Comment utiliser SPARQL explain pour analyser l'exécution des requêtes Neptune
- Exemples d'invocation de SPARQL expliqués dans Neptune

- Neptune SPARQL explique les opérateurs
- Explication des limites du SPARQL dans Neptune

Fonctionnement du moteur de requête SPARQL dans Neptune

Pour utiliser les informations fournies par la fonctionnalité SPARQL `explain`, vous devez comprendre certains points spécifiques à la façon dont le moteur de requête SPARQL Amazon Neptune fonctionne.

Le moteur convertit chaque requête SPARQL en un pipeline d'opérateurs. À partir du premier opérateur, des solutions intermédiaires appelées listes de liaisons circulent dans ce pipeline d'opérateurs. Vous pouvez vous représenter une liste de liaisons sous la forme d'une table dans laquelle les en-têtes sont un sous-ensemble des variables utilisées dans la requête. Chaque ligne de la table représente un résultat, jusqu'au point d'évaluation.

Supposons que deux préfixes d'espace de noms ont été définis pour nos données :

```
@prefix ex:   <http://example.com> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

L'exemple suivant est un exemple de liste de liaisons simple dans ce contexte :

```
?person       | ?firstName
------------------------------------------------------
ex:JaneDoe    | "Jane"
ex:JohnDoe    | "John"
ex:RichardRoe | "Richard"
```

Pour chacune des trois personnes, la liste lie la variable `?person` à un identifiant de la personne, et la variable `?firstName` à son prénom.

En règle générale, les variables peuvent rester non liées si, par exemple, il existe une sélection `OPTIONAL` d'une variable dans une requête pour laquelle aucune valeur n'est présente dans les données.

L'opérateur `PipelineJoin` est un exemple d'opérateur de moteur de requête Neptune présent dans la sortie de la fonctionnalité `explain`. Il prend en tant qu'entrée un ensemble de liaisons entrantes de l'opérateur précédent et le joint selon un modèle de triplet, par exemple `(?person, foaf:lastName, ?lastName)`. Cette opération utilise les liaisons pour la variable `?person` dans

ses flux d'entrée, les substitue dans le modèle de triplet et recherche des triplets à partir de la base de données.

Lorsqu'il est exécuté dans le contexte des liaisons entrantes de la table précédente, l'opérateur `PipelineJoin` évalue trois recherches, à savoir les éléments suivants :

```
(ex:JaneDoe,    foaf:lastName, ?lastName)
(ex:JohnDoe,    foaf:lastName, ?lastName)
(ex:RichardRoe, foaf:lastName, ?lastName)
```

Cette approche est appelée évaluation as-bound. Les solutions de ce processus d'évaluation sont jointes à nouveau sur les solutions entrantes, en remplissant le `?lastName` détecté dans les solutions entrantes. En supposant que vous trouviez un nom de famille pour les trois personnes, l'opérateur produirait une liste de liaisons sortantes qui ressemblerait à ceci :

```
?person       | ?firstName | ?lastName
---------------------------------------
ex:JaneDoe    | "Jane"     | "Doe"
ex:JohnDoe    | "John"     | "Doe"
ex:RichardRoe | "Richard"  | "Roe"
```

Cette liste de liaisons sortantes tient lieu d'entrée pour l'opérateur suivant dans le pipeline. À la fin, la sortie du dernier opérateur du pipeline définit le résultat de la requête.

Les pipelines d'opérateurs sont souvent linéaires, dans la mesure où chaque opérateur émet des solutions pour un seul opérateur connecté. Toutefois, dans certains cas, ils peuvent avoir des structures plus complexes. Par exemple, un opérateur `UNION` dans une requête SPARQL est mappé à une opération `Copy`. Cette opération duplique les liaisons et transmet la copie dans deux sous-plans, l'un pour le côté gauche et l'autre pour le côté droit de l'opérateur `UNION`.

Pour plus d'informations sur les opérateurs, consultez [Neptune SPARQL explique les opérateurs].

Comment utiliser SPARQL **`explain`** pour analyser l'exécution des requêtes Neptune

La fonctionnalité `explain` SPARQL est un outil en libre-service dans Amazon Neptune qui vous aide à comprendre l'approche d'exécution adoptée par le moteur Neptune. Pour appeler `explain`, vous transmettez un paramètre à une demande HTTP ou HTTPS dans le formulaire `explain=`*`mode`*.

Sa valeur de mode peut prendre l'une des valeurs suivantes : `static`, `dynamic` ou `details`.

- En mode statique, `explain` affiche uniquement la structure statique du plan de requête.
- En mode dynamique, `explain` inclut également les aspects dynamiques du plan de requête. Ces aspects peuvent inclure le nombre de liaisons intermédiaires transitant via les opérateurs et le ratio de liaisons sortantes par rapport aux liaisons entrantes, ainsi que le temps total pris par les opérateurs.
- En mode détails, `explain` imprime les informations affichées en mode `dynamic`, ainsi que des détails supplémentaires tels que la chaîne de requête SPARQL réelle et le nombre de plages estimé pour le modèle sous-jacent d'un opérateur de jointure.

Neptune prend en charge l'utilisation de la fonctionnalité `explain` avec les trois protocoles d'accès aux requêtes SPARQL répertoriés dans la spécification W3C SPARQL 1.1 Protocol, à savoir :

1. HTTP GET
2. HTTP POST utilisant des URL-encoded paramètres
3. HTTP POST à l'aide de paramètres de texte

Pour plus d'informations sur le moteur de requête SPARQL, consultez Fonctionnement du moteur de requête SPARQL dans Neptune.

Pour plus d'informations sur le type de résultat généré par l'appel de la fonction `explain` SPARQL, consultez Exemples d'invocation de SPARQL `expliqués` dans Neptune.

Exemples d'invocation de SPARQL **`expliqués`** dans Neptune

Les exemples de cette section montrent les différents types de sortie que vous pouvez produire en invoquant la fonctionnalité SPARQL `explain` pour analyser l'exécution des requêtes dans Amazon Neptune.

Rubriques

- Présentation de la sortie de la fonction Explain
- Exemple de sortie de mode détaillé
- Exemple de sortie de mode statique
- Différentes façons d'encoder des paramètres
- Autres types de sortie en plus text/plain
- Exemple de SPARQL expliquant la sortie lorsque le DFE est activé

Présentation de la sortie de la fonction Explain

Dans cet exemple, Jane Doe connaît deux personnes, à savoir John Doe et Richard Roe :

```
@prefix ex: <http://example.com> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

ex:JaneDoe foaf:knows ex:JohnDoe .
ex:JohnDoe foaf:firstName "John" .
ex:JohnDoe foaf:lastName "Doe" .
ex:JaneDoe foaf:knows ex:RichardRoe .
ex:RichardRoe foaf:firstName "Richard" .
ex:RichardRoe foaf:lastName "Roe" .
.
```

Pour déterminer les prénoms de toutes les personnes que Jane Doe connaît, vous pouvez écrire la requête suivante :

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -H "Accept: text/csv"
```

Cette requête simple renvoie le résultat suivant :

```
firstName
John
Richard
```

Ensuite, modifiez la commande `curl` pour appeler `explain` en ajoutant `-d "explain=dynamic"` et en utilisant le type de sortie par défaut au lieu de `text/csv` :

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=dynamic"
```

La requête renvoie maintenant la sortie dans un format ASCII bien formé (type de contenu HTTP `text/plain`), ce qui est le type de sortie par défaut :

```
#################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     # Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        # 0         # 1          # 0.00  # 0         #
#################################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        # 1         # 2          # 2.00  # 1         #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person]
            #          #          #           #       #           #
#################################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        # 2         # 2          # 1.00  # 1         #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person, ?firstName]
            #          #          #           #       #           #
#################################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   # 2         # 2          # 1.00  # 0         #
#################################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value # 2         # 2          # 1.00  # 1         #
#################################################################################################
```

Pour plus d'informations sur les opérations de la colonne `Name` et leurs arguments, consultez Opérateurs explain.

La section suivante décrit la sortie ligne par ligne :

1. La première étape de la requête principale utilise toujours l'opérateur `SolutionInjection` pour injecter une solution. La solution est ensuite étendue au résultat final via le processus d'évaluation.

   Dans ce cas, la solution soit-disant universelle `{ }` est injectée. En présence de clauses `VALUES` ou `BIND`, cette étape peut également injecter des liaisons de variable plus complexes pour commencer.

La colonne `Units Out` indique que cette solution unique découle de l'opérateur. La colonne `Out #1` spécifie l'opérateur dans lequel cet opérateur envoie le résultat. Dans cet exemple, tous les opérateurs sont connectés à l'opérateur qui suit dans la table.

2. La deuxième étape est un opérateur `PipelineJoin`. Il reçoit en tant qu'entrée la solution unique universelle (complètement sans contrainte) produite par l'opérateur précédent (`Units In := 1`). Il la joint selon le modèle de tuple défini par son argument `pattern`. Cela correspond à une recherche simple pour le modèle. Dans ce cas, le modèle de triplet est défini comme suit :

```
distinct( ex:JaneDoe, foaf:knows, ?person )
```

   L'argument `joinType := join` indique qu'il s'agit d'une jointure normale (les autres types sont notamment les jointures `optional`, `existence check`, etc.).

   L'argument `distinct := true` indique que vous extrayez uniquement les correspondances distinctes de la base de données (aucun doublon) et que vous liez les correspondances distinctes à la variable `joinProjectionVars := ?person`, dédupliquée.

   La valeur 2 de la colonne `Units Out` indique que deux solutions sont produites en sortie. Plus précisément, il s'agit des liaisons pour la variable `?person`, reflétant les deux personnes pour lesquelles les données montrent que Jane Doe les connaît :

```
?person
-------------
ex:JohnDoe
ex:RichardRoe
```

3. Les deux solutions de l'étape 2 sans transmises en tant qu'entrée (`Units In := 2`) dans le deuxième opérateur `PipelineJoin`. Cet opérateur joint les deux solutions précédentes avec le modèle de triplet suivant :

```
distinct(?person, foaf:firstName, ?firstName)
```

   Il est connu que la variable `?person` est liée à `ex:JohnDoe` ou `ex:RichardRoe` par la solution entrante de l'opérateur. Compte tenu de cela, l'opérateur `PipelineJoin` extrait les prénoms, John et Richard. Les deux solutions sortants (Units Out : = 2) se présentent alors comme suit :

```
?person       | ?firstName
---------------------------
```

```
ex:JohnDoe    | John
ex:RichardRoe | Richard
```

4. L'opérateur de projection suivante prend en tant qu'entrée les deux solutions de l'étape 3 (`Units In := 2`) et effectue une projection sur la variable `?firstName`. Cela élimine toutes les autres liaisons de variable dans les mappages et transmet les deux liaisons (`Units Out := 2`) :

   ```
   ?firstName
   ----------
   John
   Richard
   ```

5. Pour améliorer les performances, Neptune agit dans la mesure du possible au niveau des identifiants internes qu'il attribue à des termes tels que des URI et des littéraux de chaîne, plutôt que sur les chaînes proprement dites. L'opérateur final, `TermResolution`, effectue à nouveau un mappage de ces identifiants internes aux chaînes de terme correspondantes.

   Dans une évaluation de requête classique (autre que la fonction Explain), le résultat calculé par le dernier opérateur est ensuite sérialisé dans le format de sérialisation demandé et envoyé vers le client.

Exemple de sortie de mode détaillé

Supposons que vous exécutez la même requête que la précédente en mode détaillé au lieu du mode dynamique :

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=details"
```

Comme le montre cet exemple, la sortie est la même avec quelques détails supplémentaires comme la chaîne de requête en haut de la sortie et le nombre `patternEstimate` pour l'opérateur `PipelineJoin` :

```
Query:
PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://www.example.com/>
```

```
SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person foaf:firstName ?
firstName }

#################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     # Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        # 0         # 1          # 0.00  # 0         #
#################################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        # 1         # 2          # 2.00  # 13        #
#    #        #        #                   # joinType=join
            #          #         #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person]
            #          #         #           #       #           #
#    #        #        #                   # patternEstimate=2
            #          #         #           #       #           #
#################################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        # 2         # 2          # 1.00  # 3         #
#    #        #        #                   # joinType=join
            #          #         #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person, ?firstName]
            #          #         #           #       #           #
#    #        #        #                   # patternEstimate=2
            #          #         #           #       #           #
#################################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   # 2         # 2          # 1.00  # 1         #
#################################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value # 2         # 2          # 1.00  # 7         #
#################################################################################################
```

Exemple de sortie de mode statique

Supposons que vous exécutiez la même requête que la requête précédente en mode statique (mode par défaut) au lieu du mode détaillé :

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
```

```
      SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
  -d "explain=static"
```

Comme illustré dans cet exemple, la sortie est la même, si ce n'est que les trois dernières colonnes sont omises :

```
############################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     #
############################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        #
############################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        #
#    #        #        #                   # joinType=join
            #          #
#    #        #        #                   # joinProjectionVars=[?person]
            #          #
############################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        #
#    #        #        #                   # joinType=join
            #          #
#    #        #        #                   # joinProjectionVars=[?person, ?firstName]
            #          #
############################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   #
############################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value #
############################################################################################
```

Différentes façons d'encoder des paramètres

Les exemples de requête suivants illustrent deux façons différentes d'encoder des paramètres lors d'un appel de la fonction `explain` SPARQL.

Utilisation de l'encodage par URL : cet exemple utilise l'encodage de paramètres par URL et spécifie une sortie dynamique :

```
curl -XGET "http(s)://your_server:your_port/sparql?query=SELECT%20*%20WHERE%20%7B
%20%3Fs%20%3Fp%20%3Fo%20%7D%20LIMIT%20%31&explain=dynamic"
```

Spécification directe des paramètres : cette approche est identique à la requête précédente, sauf que les paramètres sont transmis via POST directement :

```
curl http(s)://your_server:your_port/sparql \
  -d "query=SELECT * WHERE { ?s ?p ?o } LIMIT 1" \
  -d "explain=dynamic"
```

Autres types de sortie en plus text/plain

Les exemples précédents utilisent le type de sortie `text/plain` par défaut. Neptune peut également formater la `explain` sortie SPARQL dans deux autres MIME-type formats, à savoir et. `text/csv` `text/html` Vous les appelez en définissant l'en-tête HTTP `Accept`, ce que vous pouvez faire à l'aide de l'indicateur `-H` dans `curl`, comme suit :

```
 -H "Accept: output type"
```

Voici quelques exemples :

Sortie **`text/csv`**

Cette requête demande une MIME-type sortie CSV en spécifiant `-H "Accept: text/csv"` :

```
curl http(s)://your_server:your_port/sparql \
  -d "query=SELECT * WHERE { ?s ?p ?o } LIMIT 1" \
  -d "explain=dynamic" \
  -H "Accept: text/csv"
```

Le format CSV, qui est utile pour l'importation dans une feuille de calcul ou une base de données, sépare les champs de chaque ligne `explain` par des points-virgules (`;`), comme suit :

```
ID;Out #1;Out #2;Name;Arguments;Mode;Units In;Units Out;Ratio;Time (ms)
0;1;-;SolutionInjection;solutions=[{}];-;0;1;0.00;0
1;2;-;PipelineJoin;pattern=distinct(?s, ?p, ?o),joinType=join,joinProjectionVars=[?s, ?
p, ?o];-;1;6;6.00;1
2;3;-;Projection;vars=[?s, ?p, ?o];retain;6;6;1.00;2
3;-;-;Slice;limit=1;-;1;1;1.00;1
```

Sortie **`text/html`**

Si vous spécifiez `-H "Accept: text/html"`, `explain` génère une table HTML :

```
<!DOCTYPE html>
<html>
  <body>
    <table border="1px">
      <thead>
        <tr>
          <th>ID</th>
          <th>Out #1</th>
          <th>Out #2</th>
          <th>Name</th>
          <th>Arguments</th>
          <th>Mode</th>
          <th>Units In</th>
          <th>Units Out</th>
          <th>Ratio</th>
          <th>Time (ms)</th>
        </tr>
      </thead>

      <tbody>
        <tr>
          <td>0</td>
          <td>1</td>
          <td>-</td>
          <td>SolutionInjection</td>
          <td>solutions=[{}]</td>
          <td>-</td>
          <td>0</td>
          <td>1</td>
          <td>0.00</td>
          <td>0</td>
        </tr>

        <tr>
          <td>1</td>
          <td>2</td>
          <td>-</td>
          <td>PipelineJoin</td>
```

```
          <td>pattern=distinct(?s, ?p, ?o)<br>
              joinType=join<br>
              joinProjectionVars=[?s, ?p, ?o]</td>
          <td>-</td>
          <td>1</td>
          <td>6</td>
          <td>6.00</td>
          <td>1</td>
        </tr>

        <tr>
          <td>2</td>
          <td>3</td>
          <td>-</td>
          <td>Projection</td>
          <td>vars=[?s, ?p, ?o]</td>
          <td>retain</td>
          <td>6</td>
          <td>6</td>
          <td>1.00</td>
          <td>2</td>
        </tr>

        <tr>
          <td>3</td>
          <td>-</td>
          <td>-</td>
          <td>Slice</td>
          <td>limit=1</td>
          <td>-</td>
          <td>1</td>
          <td>1</td>
          <td>1.00</td>
          <td>1</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

Le HTML affiche dans un navigateur une sortie similaire à ce qui suit :

ID	Out #1	Out #2	Name	Arguments	Mode	Units In	Units Out	Ratio	Time (ms)
0	1	-	SolutionInjection	solutions=[{}]	-	0	1	0.00	0
1	2	-	PipelineJoin	pattern=distinct(?s, ?p, ?o) joinType=join joinProjectionVars=[?s, ?p, ?o]	-	1	6	6.00	1
2	3	-	Projection	vars=[?s, ?p, ?o]	retain	6	6	1.00	2
3	-	-	Slice	limit=1	-	1	1	1.00	1

Exemple de SPARQL **`expliquant`** la sortie lorsque le DFE est activé

Voici un exemple de sortie SPARQL `explain` lorsque le moteur de requête alternatif Neptune DFE est activé :

```
################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments


 # Mode     # Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]


 # -        # 0        # 1         # 0.00  # 0         #
################################################################################################
# 1  # 2      # -      # HashIndexBuild    # solutionSet=solutionSet1


 # -        # 1        # 1         # 1.00  # 22        #
#    #        #        #                   # joinVars=[]


 #          #          #           #       #           #
#    #        #        #                   # sourceType=pipeline


 #          #          #           #       #           #
################################################################################################
# 2  # 3      # -      # DFENode            # DFE Stats=


      # -        # 101      # 100       # 0.99  # 32        #
#    #        #        #                   # ====> DFE execution time (measured by
 DFEQueryEngine)

     #          #          #           #       #           #
```

```
#    #        #        #                  # accepted [micros]=127


 #          #          #           #       #           #
#    #        #        #                  # ready [micros]=2


 #          #          #           #       #           #
#    #        #        #                  # running [micros]=5627


 #          #          #           #       #           #
#    #        #        #                  # finished [micros]=0


 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # ===> DFE execution time (measured in
 DFENode)

      #          #          #           #       #           #
#    #        #        #                  # -> setupTime [ms]=1


 #          #          #           #       #           #
#    #        #        #                  # -> executionTime [ms]=14


 #          #          #           #       #           #
#    #        #        #                  # -> resultReadTime [ms]=0


 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
```

```
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # ===> Static analysis statistics


 #          #          #           #       #           #
#    #        #        #                  # --> 35907 micros spent in parser.


 #          #          #           #       #           #
#    #        #        #                  # --> 7643 micros spent in range count
 estimation

      #          #          #           #       #           #
#    #        #        #                  # --> 2895 micros spent in value resolution


 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # --> 39974925 micros spent in optimizer
 loop

    #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # DFEJoinGroupNode[ children={


  #          #          #           #       #           #
#    #        #        #                  #   DFEPatternNode[(?1, TERM[117442062], ?
2, ?3) . project DISTINCT[?1, ?2] {rangeCountEstimate=100},

  #          #          #           #       #           #
```

```
#    #        #        #                  #     OperatorInfoWithAlternative[


  #          #          #           #       #           #
#    #        #        #                  #       rec=OperatorInfo[


  #          #          #           #       #           #
#    #        #        #                  #         type=INCREMENTAL_PIPELINE_JOIN,


 #          #          #           #       #           #
#    #        #        #                  #
 costEstimates=OperatorCostEstimates[

                                   #          #          #           #       #
  #
#    #        #        #                  #
 costEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0002,comp=0.0000,mem=0],

                                  #          #          #           #       #
 #
#    #        #        #                  #
 worstCaseCostEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0002,comp=0.0000,mem=0]

                       #          #          #           #       #           #
#    #        #        #                  #       alt=OperatorInfo[


  #          #          #           #       #           #
#    #        #        #                  #         type=INCREMENTAL_HASH_JOIN,


 #          #          #           #       #           #
#    #        #        #                  #
 costEstimates=OperatorCostEstimates[

                                   #          #          #           #       #
  #
#    #        #        #                  #
 costEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0003,comp=0.0000,mem=3212],

                               #          #          #           #       #           #
```

```
#    #        #        #                   #
 worstCaseCostEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0003,comp=0.0000,mem=32

                 #          #          #           #       #           #
#    #        #        #                   #   DFEPatternNode[(?1, TERM[150997262], ?
4, ?5) . project DISTINCT[?1, ?4] {rangeCountEstimate=100},

  #          #          #           #       #           #
#    #        #        #                   #     OperatorInfoWithAlternative[


  #          #          #           #       #           #
#    #        #        #                   #       rec=OperatorInfo[


  #          #          #           #       #           #
#    #        #        #                   #         type=INCREMENTAL_HASH_JOIN,


 #          #          #           #       #           #
#    #        #        #                   #
 costEstimates=OperatorCostEstimates[

                                   #          #          #           #       #
  #
#    #        #        #                   #
 costEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0003,comp=0.0000,mem=6400],

                              #          #          #           #       #           #
#    #        #        #                   #
 worstCaseCostEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0003,comp=0.0000,mem=

                 #          #          #           #       #           #
#    #        #        #                   #       alt=OperatorInfo[


  #          #          #           #       #           #
#    #        #        #                   #         type=INCREMENTAL_PIPELINE_JOIN,


 #          #          #           #       #           #
#    #        #        #                   #
 costEstimates=OperatorCostEstimates[
```

```
                                    #           #           #           #       #
  #
#    #        #        #                    #
 costEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0010,comp=0.0000,mem=0],

                                  #           #           #           #       #           #
#    #        #        #                    #
 worstCaseCostEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0010,comp=0.0000,mem=

                    #           #           #           #       #           #
#    #        #        #                    # },


 #          #          #           #       #           #
#    #        #        #                    # ]


 #          #          #           #       #           #
#    #        #        #                    #


 #          #          #           #       #           #
#    #        #        #                    # ===> DFE configuration:


 #          #          #           #       #           #
#    #        #        #                    # solutionChunkSize=5000


 #          #          #           #       #           #
#    #        #        #                    # ouputQueueSize=20


 #          #          #           #       #           #
#    #        #        #                    # numComputeCores=3


 #          #          #           #       #           #
#    #        #        #                    # maxParallelIO=10


 #          #          #           #       #           #
#    #        #        #                    # numInitialPermits=12
```

```
 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # ====> DFE configuration (reported back)


 #          #          #           #       #           #
#    #        #        #                   # numComputeCores=3


 #          #          #           #       #           #
#    #        #        #                   # maxParallelIO=2


 #          #          #           #       #           #
#    #        #        #                   # numInitialPermits=12


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # ===> Statistics & operator histogram


 #          #          #           #       #           #
#    #        #        #                   # ==> Statistics


 #          #          #           #       #           #
#    #        #        #                   # -> 3741 / 3668 micros total elapsed (incl.
 wait / excl. wait)

 #          #          #           #       #           #
#    #        #        #                   # -> 3741 / 3 millis total elapse (incl.
 wait / excl. wait)
```

```
    #          #          #           #       #           #
#    #        #        #                  # -> 3741 / 0 secs total elapsed (incl.
 wait / excl. wait)

     #          #          #           #       #           #
#    #        #        #                  # ==> Operator histogram


 #          #          #           #       #           #
#    #        #        #                  # -> 47.66% of total time (excl. wait):
 pipelineScan (2 instances)

     #          #          #           #       #           #
#    #        #        #                  # -> 10.99% of total time (excl. wait):
 merge (1 instances)

     #          #          #           #       #           #
#    #        #        #                  # -> 41.17% of total time (excl. wait):
 symmetricHashJoin (1 instances)

     #          #          #           #       #           #
#    #        #        #                  # -> 0.19% of total time (excl. wait): drain
 (1 instances)

 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # nodeId | out0   | out1 | opName
   | args                                          | rowsIn | rowsOut | chunksIn |
 chunksOut | elapsed* | outWait | outBlocked | ratio    | rate* [M/s] | rate [M/s] | %
    #          #          #           #       #           #
#    #        #        #                  # ------ | ------ | ---- | -----------------
 | ------------------------------------------------ | ------ | ------- | -------- |
 --------- | -------- | ------- | ---------- | -------- | ----------- | ---------- |
 ----- #          #          #           #       #           #
#    #        #        #                  # node_0 | node_2 | -    | pipelineScan
  | (?1, TERM[117442062], ?2, ?3) DISTINCT [?1, ?2]  | 0      | 100     | 0        | 1
        | 874      | 0       | 0          | Infinity | 0.1144      | 0.1144     | 23.83
 #          #          #           #       #           #
#    #        #        #                  # node_1 | node_2 | -    | pipelineScan
  | (?1, TERM[150997262], ?4, ?5) DISTINCT [?1, ?4]  | 0      | 100     | 0        | 1
```

```
        | 874      | 0       | 0          | Infinity | 0.1144      | 0.1144     | 23.83
 #          #          #           #       #           #
#    #        #        #                   # node_2 | node_4 | -    | symmetricHashJoin
 |                                                    | 200    | 100     | 2        | 2
        | 1510     | 73      | 0          | 0.50     | 0.0662      | 0.0632     | 41.17
 #          #          #           #       #           #
#    #        #        #                   # node_3 | -      | -    | drain
  |                                                   | 100    | 0       | 1        | 0
        | 7        | 0       | 0          | 0.00     | 0.0000      | 0.0000     | 0.19
 #          #          #           #       #           #
#    #        #        #                   # node_4 | node_3 | -    | merge
  |                                                   | 100    | 100     | 2        | 1
        | 403      | 0       | 0          | 1.00     | 0.2481      | 0.2481     | 10.99
 #          #          #           #       #           #
#########################################################################################
# 3  # 4      # -      # HashIndexJoin     # solutionSet=solutionSet1


 # -        # 100      # 100       # 1.00  # 4         #
#    #        #        #                   # joinType=join


 #          #          #           #       #           #
#########################################################################################
# 4  # 5      # -      # Distinct          # vars=[?s, ?o, ?o1]


 # -        # 100      # 100       # 1.00  # 9         #
#########################################################################################
# 5  # 6      # -      # Projection        # vars=[?s, ?o, ?o1]


 # retain   # 100      # 100       # 1.00  # 2         #
#########################################################################################
# 6  # -      # -      # TermResolution    # vars=[?s, ?o, ?o1]


 # id2value # 100      # 100       # 1.00  # 11        #
#########################################################################################
```

Neptune SPARQL explique les opérateurs

Les sections suivantes décrivent les opérateurs et les paramètres de la fonctionnalité SPARQL `explain` actuellement disponibles dans Amazon Neptune.

> **Important**
>
> La fonction `explain` SPARQL est encore en cours d'affinement. Les opérateurs et paramètres documentés ici sont susceptibles de changer dans de futures versions.

Rubriques

- Opérateur d'agrégation
- Opérateur ConditionalRouting
- Opérateur de copie
- opérateur DFEnode
- Opérateur distinct
- Opérateur de fédération
- Opérateur du filtre
- Opérateur HashIndexBuild
- Opérateur HashIndexJoin
- Opérateur MergeJoin
- Opérateur NamedSubquery
- Opérateur PipelineJoin
- Opérateur PipelineCountJoin
- Opérateur PipelinedHashIndexJoin
- Opérateur de projection
- Opérateur PropertyPath
- Opérateur TermResolution
- Opérateur Slice
- Opérateur SolutionInjection
- Opérateur de tri

- Opérateur VariableAlignment

Opérateur **`d'agrégation`**

Effectue une ou plusieurs agrégations, en implémentant la sémantique des opérateurs d'agrégation SPARQL tels que `count`, `max`, `min`, `sum`, etc.

`Aggregation` est fourni avec un regroupement facultatif à l'aide de clauses `groupBy` et de contraintes `having` facultatives.

Arguments

- `groupBy` : (facultatif) fournit une clause `groupBy` qui spécifie la séquence d'expressions selon laquelle les solutions entrantes sont regroupées.
- `aggregates` : (obligatoire) spécifie une liste ordonnée d'expressions d'agrégation.
- `having` : (facultatif) ajoute des contraintes pour filtrer des groupes comme impliqué par la clause `having` dans la requête SPARQL.

Opérateur **`ConditionalRouting`**

Achemine des solutions entrants en fonction d'une condition. Les solutions qui remplissent les conditions sont acheminées vers l'ID d'opérateur référencé par `Out #1`, tandis que les solutions qui ne les remplissent pas sont acheminées vers l'opérateur référencé par `Out #2`.

Arguments

- `condition` : (obligatoire) condition de routage.

Opérateur de **`copie`**

Délègue le flux de solution comme spécifié par le mode indiqué.

Modes

- `forward` : transmet les solutions à l'opérateur en aval identifié par `Out #1`.
- `duplicate` : duplique les solutions et les transmet à chacun des deux opérateurs identifiées par `Out #1` et `Out #2`.

`Copy` ne comporte aucun argument.

opérateur DFEnode

Cet opérateur est une abstraction du plan exécuté par le moteur de requête alternatif DFE. Le plan DFE détaillé est décrit dans les arguments de cet opérateur. L'argument est actuellement surchargé pour contenir les statistiques d'exécution détaillées du plan DFE. Il contient le temps consacré aux différentes étapes de l'exécution des requêtes par le DFE.

L'arbre syntaxique abstrait optimisé (AST) logique pour le plan de requêtes DFE est imprimé avec des informations sur les types d'opérateurs pris en compte lors de la planification et les coûts les plus et les moins favorables associés à l'exécution des opérateurs. L'arbre AST comprend actuellement les types de nœuds suivants :

- `DFEJoinGroupNode` : représente une jointure d'un ou de plusieurs éléments `DFEPatternNodes`.
- `DFEPatternNode` : encapsule un modèle sous-jacent à l'aide duquel les tuples correspondants sont projetés hors de la base de données sous-jacente.

La sous-section, `Statistics & Operator histogram`, contient des détails sur le temps d'exécution du plan `DataflowOp` et sur la répartition du temps CPU utilisé par chaque opérateur. En dessous se trouve un tableau qui affiche les statistiques d'exécution détaillées du plan exécuté par le DFE.

> **Note**
>
> Le support DFE pour SPARQL étant une fonctionnalité expérimentale, le format exact de sa `explain` sortie peut changer.

Opérateur distinct

Calcule la projection distincte sur un sous-ensemble des variables, en éliminant les doublons. Par conséquent, le nombre de solutions en entrée est supérieur ou égal au nombre de solutions en sortie.

Arguments

- `vars` : (obligatoire) variables auxquelles appliquer la projection `Distinct`.

Opérateur de fédération

Transmet une requête spécifique à un point de terminaison SPARQL distant spécifique.

Arguments

- `SERVICE` : (obligatoire) URL du point de terminaison dans la déclaration SPARQL `endpoint`. Il peut s'agir d'une chaîne constante, ou, si le point de terminaison de la requête est déterminé en fonction d'une variable au sein de la même requête, il peut s'agir du nom de la variable.
- `query` : (obligatoire) chaîne de requête reconstruite à envoyer au point de terminaison distant. Le moteur ajoute des préfixes par défaut à cette requête, même lorsque le client n'en spécifie aucun.
- `silent` : (obligatoire) valeur booléenne qui indique si le mot-clé `SILENT` apparaît après le mot-clé. `SILENT` indique au moteur de ne pas faire échouer l'ensemble de la requête, même si la partie `SERVICE` distante échoue.

Opérateur du **`filtre`**

Filtre les solutions entrantes. Seules les solutions qui remplissent la condition de filtre sont transmises à l'opérateur en amont. Toutes les autres sont supprimées.

Arguments

- `condition` : (obligatoire) condition de filtre.

Opérateur **`HashIndexBuild`**

Prend une liste de liaisons et les place dans un index de hachage dont le nom est défini par l'argument `solutionSet`. En général, les opérateurs suivants effectuent des jointures en fonction de cette solution, en y faisant référence par ce nom.

Arguments

- `solutionSet` : (obligatoire) nom de l'ensemble de solutions de l'index de hachage.
- `sourceType` : (obligatoire) type de la source à partir de laquelle les liaisons à stocker dans l'index de hachage sont obtenues :
  - `pipeline` : place les solutions entrantes de l'opérateur en aval dans le pipeline des opérateurs dans l'index de hachage.
  - `binding set` : place l'ensemble de liaisons fixes spécifié par l'argument `sourceBindingSet` dans l'index de hachage.
- `sourceBindingSet` : (facultatif) si la valeur de l'argument `sourceType` est `binding set`, cet argument spécifie l'ensemble de liaisons statiques à placer dans l'index de hachage.

Opérateur **HashIndexJoin**

Joint les solutions entrantes par rapport à l'ensemble de solutions de l'index de hachage, identifié par l'argument `solutionSet`.

Arguments

- `solutionSet` : (obligatoire) nom de l'ensemble de solutions sur lequel effectuer la jointure. Il doit s'agir d'un index de hachage qui a été créé dans une étape précédente à l'aide de l'opérateur `HashIndexBuild`.
- `joinType` : (obligatoire) type de jointure à effectuer :
  - `join` : jointure normale, nécessitant une correspondance exacte entre toutes les variables partagées.
  - `optional` : jointure `OPTIONAL` qui utilise la sémantique de l'opérateur `optional` SPARQL.
  - `minus` : une opération `MINUS` conserve un mappage pour lequel aucun partenaire de jointure n'existe, à l'aide de la sémantique de l'opérateur `minus` SPARQL.
  - `existence check` : vérifie s'il existe ou non un partenaire de jointure et lie la variable `existenceCheckResultVar` au résultat de cette vérification.
- `constraints` : (facultatif) des contraintes de jointure supplémentaires sont prises en compte lors de la jointure. Les jointures qui ne remplissent pas ces contraintes sont rejetées.
- `existenceCheckResultVar` : (facultatif) utilisé uniquement pour les jointures où `joinType` est égal à `existence check` (voir l'argument `joinType` ci-dessus).

Opérateur **MergeJoin**

Jointure par fusion sur plusieurs ensembles de solutions, identifiés par l'argument `solutionSets`.

Arguments

- `solutionSets` : (obligatoire) ensembles de solutions à joindre.

Opérateur **NamedSubquery**

Déclenche l'évaluation de la sous-requête identifiée par l'argument `subQuery` et place le résultat dans l'ensemble de solutions spécifié par l'argument `solutionSet`. Les solutions entrantes pour l'opérateur sont transmises à la sous-requête, puis à l'opérateur suivant.

Arguments

- `subQuery` : (obligatoire) nom de la sous-requête à évaluer. La sous-requête est affichée explicitement dans la sortie.
- `solutionSet` : (obligatoire) nom de l'ensemble de solutions dans lequel vous souhaitez stocker le résultat de la sous-requête.

Opérateur **`PipelineJoin`**

Reçoit en tant qu'entrée la sortie de l'opérateur précédent et la joint en fonction du modèle de tuple défini par l'argument `pattern`.

Arguments

- `pattern` : (obligatoire) modèle, qui se présente sous la forme d'un sujet-prédicat-objet et, le cas échéant, du tuple -graph qui sous-tend la jointure. Si `distinct` est spécifié pour le modèle, la jointure extrait uniquement les solutions distinctes des variables de projection spécifiées par l'argument `projectionVars`, et non toutes les solutions qui correspondent.
- `inlineFilters` : (facultatif) ensemble de filtres à appliquer aux variables dans le modèle. Le modèle est évalué conjointement avec ces filtres.
- `joinType` : (obligatoire) type de jointure à effectuer :
  - `join` : jointure normale, nécessitant une correspondance exacte entre toutes les variables partagées.
  - `optional` : jointure `OPTIONAL` qui utilise la sémantique de l'opérateur `optional` SPARQL.
  - `minus` : une opération `MINUS` conserve un mappage pour lequel aucun partenaire de jointure n'existe, à l'aide de la sémantique de l'opérateur `minus` SPARQL.
  - `existence check` : vérifie s'il existe ou non un partenaire de jointure et lie la variable `existenceCheckResultVar` au résultat de cette vérification.
- `constraints` : (facultatif) des contraintes de jointure supplémentaires sont prises en compte lors de la jointure. Les jointures qui ne remplissent pas ces contraintes sont rejetées.
- `projectionVars` : (facultatif) variables de la projection. Utilisé en combiné avec `distinct := true` pour appliquer l'extraction de projections distinctes sur un ensemble de variables spécifié.
- `cutoffLimit` : (facultatif) limite de coupure pour le nombre de partenaires de jointure extraits. Même s'il n'existe pas de limite par défaut, vous pouvez définir cet argument sur 1 lorsque vous effectuez des jointures pour implémenter des clauses `FILTER (NOT) EXISTS`, où il suffit de prouver ou de réfuter qu'il existe une jointure partenaire.

Opérateur **PipelineCountJoin**

Variante de `PipelineJoin`. Au lieu d'effectuer une jointure, compte uniquement les partenaires de jointure correspondants et lie le comptage à la variable spécifiée par l'argument `countVar`.

Arguments

- `countVar` : (obligatoire) variable à laquelle le résultat du décompte, à savoir le nombre de partenaires de jointure, doit être limitée.
- `pattern` : (obligatoire) modèle, qui se présente sous la forme d'un sujet-prédicat-objet et, le cas échéant, du tuple -graph qui sous-tend la jointure. Si `distinct` est spécifié pour le modèle, la jointure extrait uniquement les solutions distinctes des variables de projection spécifiées par l'argument `projectionVars`, et non toutes les solutions qui correspondent.
- `inlineFilters` : (facultatif) ensemble de filtres à appliquer aux variables dans le modèle. Le modèle est évalué conjointement avec ces filtres.
- `joinType` : (obligatoire) type de jointure à effectuer :
  - `join` : jointure normale, nécessitant une correspondance exacte entre toutes les variables partagées.
  - `optional` : jointure `OPTIONAL` qui utilise la sémantique de l'opérateur `optional` SPARQL.
  - `minus` : une opération `MINUS` conserve un mappage pour lequel aucun partenaire de jointure n'existe, à l'aide de la sémantique de l'opérateur `minus` SPARQL.
  - `existence check` : vérifie s'il existe ou non un partenaire de jointure et lie la variable `existenceCheckResultVar` au résultat de cette vérification.
- `constraints` : (facultatif) des contraintes de jointure supplémentaires sont prises en compte lors de la jointure. Les jointures qui ne remplissent pas ces contraintes sont rejetées.
- `projectionVars` : (facultatif) variables de la projection. Utilisé en combiné avec `distinct := true` pour appliquer l'extraction de projections distinctes sur un ensemble de variables spécifié.
- `cutoffLimit` : (facultatif) limite de coupure pour le nombre de partenaires de jointure extraits. Même s'il n'existe pas de limite par défaut, vous pouvez définir cet argument sur 1 lorsque vous effectuez des jointures pour implémenter des clauses `FILTER (NOT) EXISTS`, où il suffit de prouver ou de réfuter qu'il existe une jointure partenaire.

Opérateur **PipelinedHashIndexJoin**

Il s'agit d'un index de hachage et d'un opérateur de jointure tout-en-un. Il utilise une liste de liaisons, les regroupe dans un index de hachage, puis joint les solutions entrantes à l'index de hachage.

Arguments

- `sourceType` : (obligatoire) type de la source à partir de laquelle les liaisons à stocker dans l'index de hachage sont obtenues :
  - `pipeline` : incite `PipelinedHashIndexJoin` à placer les solutions entrantes de l'opérateur en aval dans le pipeline des opérateurs dans l'index de hachage.
  - `binding set` : incite `PipelinedHashIndexJoin` à place l'ensemble de liaisons fixes spécifié par l'argument `sourceBindingSet` dans l'index de hachage.
- `sourceSubQuery` : (facultatif) si la valeur de l'argument `sourceType` est `pipeline`, cet argument indique la sous-requête qui est évaluée et insérée dans l'index de hachage.
- `sourceBindingSet` : (facultatif) si la valeur de l'argument `sourceType` est `binding set`, cet argument spécifie l'ensemble de liaisons statiques à placer dans l'index de hachage.
- `joinType` : (obligatoire) type de jointure à effectuer :
  - `join` : jointure normale, nécessitant une correspondance exacte entre toutes les variables partagées.
  - `optional` : jointure `OPTIONAL` qui utilise la sémantique de l'opérateur `optional` SPARQL.
  - `minus` : une opération `MINUS` conserve un mappage pour lequel aucun partenaire de jointure n'existe, à l'aide de la sémantique de l'opérateur `minus` SPARQL.
  - `existence check` : vérifie s'il existe ou non un partenaire de jointure et lie la variable `existenceCheckResultVar` au résultat de cette vérification.
- `existenceCheckResultVar` : (facultatif) utilisé uniquement pour les jointures où `joinType` est égal à `existence check` (voir l'argument joinType ci-dessus).

Opérateur **`de projection`**

Effectue une projection sur un sous-ensemble des variables. Le nombre de solutions en entrée est égal au nombre de solutions en sortie, mais la forme de la solution diffère selon le paramètre de mode.

Modes

- `retain` : ne conserve dans les solutions que les variables qui sont spécifiées par l'argument `vars`.
- `drop` : abandonne toutes les variables qui sont spécifiées par l'argument `vars`.

Arguments

- `vars` : (obligatoire) variables à conserver ou à abandonner, en fonction du paramètre de mode.

Opérateur **`PropertyPath`**

Active les chemins de propriétés récursifs tels que `+` ou `*`. Neptune met en œuvre une approche d'itération à point fixe basée sur un modèle spécifié par l'argument `iterationTemplate`. Les variables de gauche ou de droite connues sont liées dans le modèle pour chaque itération de type point fixe, jusqu'à ce qu'aucune autre nouvelle solution ne puisse être trouvée.

Arguments

- `iterationTemplate` : (obligatoire) nom du modèle de sous-requête utilisé pour implémenter l'itération de type point fixe.
- `leftTerm` : (obligatoire) terme (variable ou constante) à gauche du chemin de propriété.
- `rightTerm` : (obligatoire) terme (variable ou constante) à droite du chemin de propriété.
- `lowerBound` : (obligatoire) limite inférieure pour une itération de type point fixe (`0` pour les requêtes `*` ou 1 pour les requêtes `+`).

Opérateur **`TermResolution`**

Reconvertit des valeurs d'identifiant de chaîne interne en leurs chaînes externes correspondantes, ou convertit des chaînes externes en valeurs d'identifiant de chaîne interne, en fonction du mode.

Modes

- `value2id` : mappe des termes tels que des littéraux et des URI avec les valeurs d'ID internes correspondantes (encodage en valeurs internes).
- `id2value` : mappe des valeurs d'ID internes avec les termes correspondants tels que des littéraux et des URI (décodage de valeurs internes).

Arguments

- `vars` : (obligatoire) spécifie les variables dont les chaînes ou les ID de chaîne interne doivent être mappés.

Opérateur **`Slice`**

Implémente une tranche sur le flux de solution entrant, à l'aide de la sémantique des clauses `LIMIT` et `OFFSET` SPARQL.

Arguments

- `limit` : (facultatif) limite applicable aux solutions à transmettre.
- `offset` : (facultatif) décalage par rapport auquel les solutions sont évaluées pour la transmission.

Opérateur **`SolutionInjection`**

Ne reçoit aucune entrée. Injecte statiquement des solutions dans le plan de requête et les enregistre dans l'argument `solutions`.

Les plans de requête commencent toujours par cette injection statique. Si des solutions statiques à injecter peuvent être dérivées de la requête proprement dite en combinant différentes sources de liaisons statiques (par exemple, à partir de clauses `VALUES` ou `BIND`), l'opérateur `SolutionInjection` injecte ces solutions statiques dérivées. Dans le cas le plus simple, elles reflètent des liaisons qui sont impliquées par une clause `VALUES` externe.

Si aucune solution statique ne peut être dérivée à partir de la requête, `SolutionInjection` injecte la solution soit-disant universelle vide qui est étendue et multipliée tout au long du processus d'évaluation de requête.

Arguments

- `solutions` : (obligatoire) séquence de solutions injectées par l'opérateur.

Opérateur **`de tri`**

Trie l'ensemble de solutions à l'aide des conditions de tri spécifiées.

Arguments

- `sortOrder` : (obligatoire) liste ordonnée de variables, chacune contenant un élément `ASC` (croissant) ou `DESC` (décroissant), utilisées de manière séquentielle pour trier l'ensemble de solutions.

Opérateur **`VariableAlignment`**

Inspecte les solutions une par une, en alignant chacune d'entre elles sur deux variables : une variable `sourceVar` spécifiée et une variable `targetVar` spécifiée.

Si `sourceVar` et `targetVar` dans une solution ont la même valeur, les variables sont considérées comme étant alignées et la solution est transmise avec les variables `sourceVar` redondantes sont projetées.

Si les variables sont liées à des valeurs différentes, la solution est filtrée entièrement.

Arguments

- `sourceVar` : (obligatoire) variable source à comparer à la variable cible. Si l'alignement réussit dans une solution, ce qui signifie que les deux variables ont la même valeur, la variable source est projetée.
- `targetVar` : (obligatoire) variable cible à laquelle la variable source est comparée. Elle est conservée même lorsque l'alignement réussit.

Explication des limites du SPARQL dans Neptune

La version de la fonction SPARQL `explain` dans Neptune présente les limites suivantes.

Neptune ne prend actuellement en charge Explain que dans les requêtes SELECT SPARQL

Pour plus d'informations sur le processus d'évaluation pour d'autres formes de requêtes, comme les requêtes `ASK`, `CONSTRUCT`, `DESCRIBE` et `SPARQL UPDATE`, vous pouvez transformer ces requêtes en une requête SELECT. Utilisez ensuite `explain` pour inspecter la requête SELECT correspondante.

Par exemple, pour obtenir des informations `explain` sur une requête `ASK WHERE {...}`, exécutez la requête `SELECT WHERE {...} LIMIT 1` correspondante avec `explain`.

De même, pour une requête `CONSTRUCT {...} WHERE {...}`, supprimez la partie `CONSTRUCT {...}` et exécutez une requête `SELECT` avec `explain` sur la deuxième clause `WHERE {...}`. L'évaluation de la deuxième clause `WHERE` révèle généralement les principales difficultés du traitement de la requête `CONSTRUCT`, car les solutions découlant de la deuxième clause `WHERE` dans le modèle `CONSTRUCT` ne nécessitent généralement qu'une substitution simple.

Les opérateurs de la fonction Explain peuvent changer dans des versions futures

Les opérateurs de la fonction `explain` SPARQL et leurs paramètres sont susceptibles d'être modifiés dans des versions futures.

La sortie de la fonction Explain peut changer dans des versions futures

Par exemple, des en-têtes de colonne peuvent être modifiés et d'autres colonnes peuvent être ajoutées aux tables.

Requêtes fédérées SPARQL dans Neptune à l'aide de l'extension SERVICE

Amazon Neptune prend entièrement en charge l'extension de requête fédérée SPARQL qui utilise le mot-clé `SERVICE`. (Pour plus d'informations, consultez Requête fédérée SPARQL 1.1.)

Le mot-clé `SERVICE` indique au moteur de requête SPARQL qu'il doit exécuter une partie de la requête sur un point de terminaison SPARQL distant et composer le résultat de la requête finale. Seules les opérations `READ` sont possibles. Les opérations `WRITE` et `DELETE` ne sont pas prises en charge. Neptune ne peut exécuter des requêtes fédérées que sur des points de terminaison SPARQL accessibles au sein de son cloud privé virtuel (VPC). Toutefois, vous pouvez également utiliser un proxy inverse dans le VPC pour rendre une source de données externe accessible au sein du VPC.

> **Note**
>
> Lorsque SPARQL `SERVICE` est utilisé pour fédérer une requête à deux clusters Neptune ou plus dans le même VPC, les groupes de sécurité doivent être configurés pour autoriser tous ces clusters Neptune à communiquer entre eux.

> **Important**
>
> La fonction SPARQL 1.1 Federation effectue des demandes de service en votre nom lorsqu'elle transmet des requêtes et des paramètres à des points de terminaison SPARQL externes. Il vous incombe de vérifier que les points de terminaison SPARQL externes répondent aux exigences de sécurité et de gestion des données de votre application.

Exemple de requête fédérée Neptune

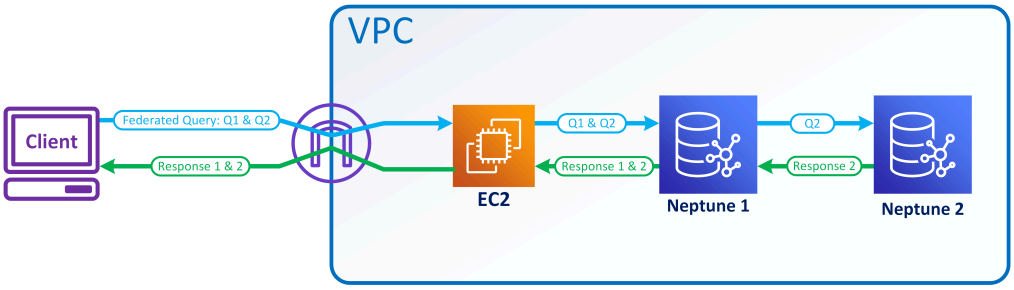
L'exemple simple suivant montre le fonctionnement des requêtes fédérées SPARQL.

Supposons qu'un client envoie la requête suivante à Neptune-1at`http://neptune-1:8182/sparql`.

```
SELECT * WHERE {
   ?person rdf:type foaf:Person .
   SERVICE <http://neptune-2:8182/sparql> {
       ?person foaf:knows ?friend .
    }
}
```

1. Neptune-1évalue le premier modèle de requête (Q-1)`?person rdf:type foaf:Person`, c'est-à-dire utilise les résultats pour résoudre `?person` in Q-2(`?person foaf:knows ?friend`), et transmet le modèle résultant à Neptune-2at`http://neptune-2:8182/sparql`.
2. Neptune-2évalue Q-2et renvoie les résultats à. Neptune-1
3. Neptune-1associe les solutions pour les deux modèles et renvoie les résultats au client.

Ce flux est illustré dans le diagramme suivant.



> **Note**
>
> « Par défaut, l'optimiseur détermine à quel moment de l'exécution de la requête la déclaration `SERVICE` est exécutée. Vous pouvez annuler ce placement à l'aide de l'indicateur de requête joinOrder.

Contrôle d'accès pour les requêtes fédérées dans Neptune

Neptune utilise Gestion des identités et des accès AWS (IAM) pour l'authentification et l'autorisation. Le contrôle d'accès d'une requête fédérée peut impliquer plusieurs instances de base de données Neptune. Ces instances peuvent avoir des exigences différentes pour le contrôle d'accès. Dans certaines circonstances, cela peut limiter votre capacité à effectuer une requête fédérée.

Prenons l'exemple simple présenté dans la section précédente. Neptune-1appels Neptune-2avec les mêmes informations d'identification avec lesquelles il a été appelé.

- Si l'authentification et l'autorisation IAM sont Neptune-1requises, mais Neptune-2que ce n'est pas le cas, vous n'avez besoin que des autorisations IAM appropriées Neptune-1pour effectuer la requête fédérée.
- Si Neptune-1Neptune-2les deux nécessitent une authentification et une autorisation IAM, vous devez associer des autorisations IAM aux deux bases de données pour effectuer la requête fédérée. Les deux clusters doivent également se trouver dans le même AWS compte et dans la même région. Cross-region and/or les architectures de requêtes fédérées entre comptes ne sont actuellement pas prises en charge.
- Toutefois, dans le cas où ce n'Neptune-1est pas le cas IAM-enabled mais l'Neptune-2est, vous ne pouvez pas effectuer de requête fédérée. La raison en est que Neptune-1vous ne pouvez pas récupérer vos informations d'identification IAM et les transmettre Neptune-2pour autoriser la deuxième partie de la requête.

Outils de visualisation de graphe pour Neptune

Outre les fonctionnalités de visualisation intégrées aux carnets graphiques Neptune, vous pouvez également utiliser des solutions conçues par des AWS partenaires et des fournisseurs tiers pour visualiser les données stockées dans Neptune.

La visualisation sophistiquée des graphes aide les scientifiques des données, les dirigeants et les autres acteurs d'une organisation à explorer les données de graphe de manière interactive, sans avoir à savoir comment écrire des requêtes complexes.

Rubriques

- Outil graph-explorer open source
- Logiciel Tom Sawyer
- Cambridge Intelligence
- Graphistry
- metaphacts
- G.V() client de base de données de graphes
- Linkurious
- Graph.Build

Outil graph-explorer open source

Graph-explorerest un outil open source d'exploration visuelle low-code pour les données graphiques, disponible sous licence. Apache-2.0 Il vous permet de parcourir les données des graphes de propriétés étiquetés (LPG) ou du Resource Description Framework (RDF) dans une base de données de graphes sans avoir à écrire de requêtes graphiques. Graph-explorer est destiné à aider les scientifiques des données, les analystes commerciaux et les autres acteurs d'une organisation à explorer les données graphiques de manière interactive sans avoir à apprendre un langage de requête graphique.

Graph-explorer fournit une application React-based Web qui peut être déployée en tant que conteneur pour visualiser les données graphiques. Vous pouvez vous connecter à Amazon Neptune ou à d'autres bases de données graphiques qui fournissent un point de terminaison Apache TinkerPop G705 ou SPARQL 1.1.

- Vous pouvez consulter rapidement un résumé des données à l'aide des filtres à facettes, ou effectuer une recherche dans les données en saisissant du texte dans la barre de recherche.
- Vous pouvez également explorer de manière interactive les connexions des nœuds et des arêtes. Vous pouvez visualiser les voisins des nœuds pour déterminer comment les objets sont liés les uns aux autres, puis effectuer une exploration plus approfondie pour inspecter visuellement les arêtes et les propriétés.
- Vous pouvez également personnaliser la mise en page du graphe, les couleurs, les icônes et les propriétés par défaut à afficher pour les nœuds et les arêtes. Pour les graphes RDF, vous pouvez également personnaliser les espaces de noms pour les URI des ressources.
- Pour les rapports et les présentations impliquant des données de graphe, vous pouvez configurer et enregistrer les vues que vous avez créées dans un format PNG haute résolution. Vous pouvez également télécharger les données associées dans un fichier CSV ou JSON pour un traitement ultérieur.

Utilisation de l'explorateur de graphes dans un bloc-notes Neptune

Le moyen le plus simple d'utiliser l'explorateur de graphes avec Neptune est d'utiliser un bloc-notes de graphes Neptune.

Si vous utilisez Neptune Workbench pour héberger un bloc-notes Neptune, l'explorateur de graphes est automatiquement déployé avec le bloc-notes et connecté à Neptune.

Après avoir créé un bloc-notes, accédez à la console Neptune pour démarrer l'explorateur de graphes :

1. Accédez à Neptune.
2. Sous Bloc-notes, sélectionnez votre bloc-notes.
3. Sous Actions, choisissez Ouvrir l'explorateur de graphes.

Comment exécuter Graph-Explorer dans Amazon ECS sur AWS Fargate et connectez-vous à Neptune

Vous pouvez également créer l'image Docker de l'explorateur de graphes et l'exécuter sur une machine locale ou un service hébergé tel qu'Amazon Elastic Compute Cloud (Amazon EC2) ou Amazon Elastic ContainerService (Amazon ECS), comme expliqué dans la section Getting Started du document read-me in the graph-explorer. GitHub

À titre d'exemple, la section suivante fournit des instructions pas à pas pour l'exécution de l'explorateur de graphes dans Amazon ECS sur AWS Fargate :

1. Créez un rôle IAM et attachez-y la nouvelle politique.

   - AmazonECSTaskExecutionRolePolicy
   - CloudWatchLogsFullAccess

   Conservez le nom du rôle à portée de main pour pouvoir l'utiliser dans un instant.

2. Créez un cluster Amazon ECS avec l'infrastructure définie sur FARGATE et les options réseau suivantes :

   - `VPC` : défini sur le VPC où se trouve votre base de données Neptune.
   - `Subnets` : défini sur les sous-réseaux publics de ce VPC (supprimez tous les autres).

3. Créez une définition de tâche JSON comme suit :

```
{
  "family": "explorer-test",
  "containerDefinitions": [
    {
      "name": "graph-explorer",
      "image": "public.ecr.aws/neptune/graph-explorer:latest",
      "cpu": 0,
      "portMappings": [
        {
          "name": "graph-explorer-80-tcp",
          "containerPort": 80,
          "hostPort": 80,
          "protocol": "tcp",
          "appProtocol": "http"
        },
        {
          "name": "graph-explorer-443-tcp",
          "containerPort": 443,
          "hostPort": 443,
          "protocol": "tcp",
          "appProtocol": "http"
        }
      ],
      "essential": true,
```

```
      "environment": [
        {
          "name": "HOST",
          "value": "localhost"
        }
      ],
      "mountPoints": [],
      "volumesFrom": [],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "true",
          "awslogs-group": "/ecs/graph-explorer",
          "awslogs-region": "{region}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    }
  ],
  "taskRoleArn": "arn:aws:iam::{account_no}:role/{role_name_from_step_1}",
  "executionRoleArn": "arn:aws:iam::{account_no}:role/{role_name_from_step_1}",
  "networkMode": "awsvpc",
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "cpu": "1024",
  "memory": "3072",
  "runtimePlatform": {
    "cpuArchitecture": "X86_64",
    "operatingSystemFamily": "LINUX"
  }
}
```

4. Démarrez une nouvelle tâche en utilisant les paramètres par défaut, à l'exception des champs suivants :

   - Environnement
     - Options de calcul => Type de lancement
   - Configuration de déploiement
     - Type d'application => Tâche
     - Famille => *(your new JSON task definition)*

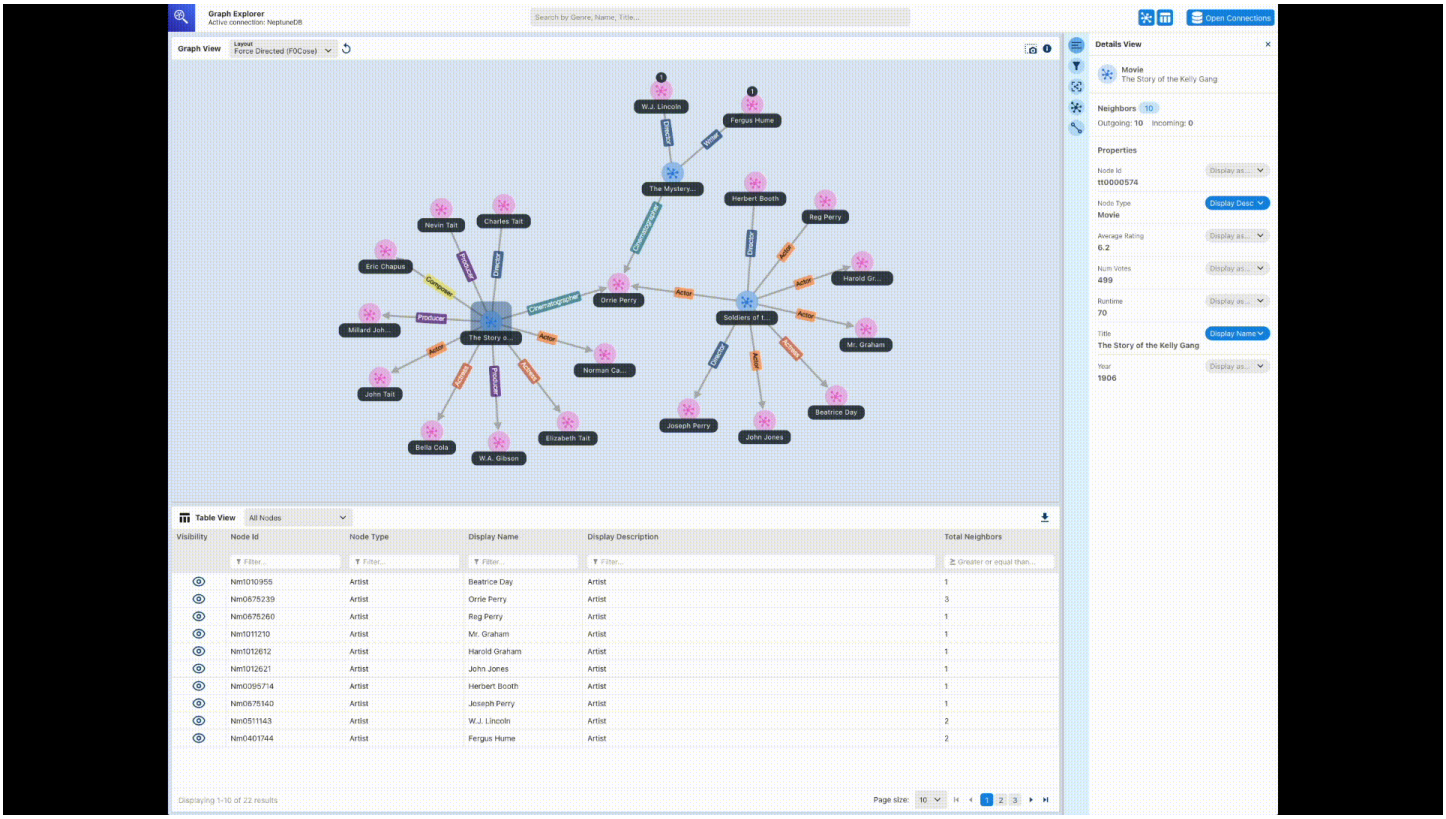
        - Révision => *(latest)*
    - Réseaux
        - VPC => *(the Neptune VPC you want to connect to)*
        - Sous-réseaux => *(ONLY the public subnets of the VPC– remove all others)*
        - Groupe de sécurité => Créer un groupe de sécurité
        - Nom du groupe de sécurité => graph-explorer
        - Description du groupe de sécurité = Groupe de sécurité pour l'accès à graph-explorer
        - Règles entrantes pour les groupes de sécurité =>
            1. 80 Anywhere
            2. 443 Anywhere
5. Sélectionnez Créer.
6. Une fois la tâche démarrée, copiez l'adresse IP publique de la tâche en cours d'exécution et accédez à : `https://(your public IP)/explorer`.
7. Acceptez le risque lié à l'utilisation du certificat non reconnu qui a été généré ou ajoutez-le à votre chaîne de clé.
8. Vous pouvez désormais ajouter une connexion à Neptune. Créez une connexion soit pour un graphe de propriétés (LPG), soit pour RDF, puis définissez les champs suivants :

```
Using proxy server => true
Public or Proxy Endpoint => https://(your public IP address)
Graph connection URL => https://(your Neptune endpoint):8182
```

Vous devriez maintenant être connecté.

Graph-explorer démonstration

Cette courte vidéo vous donne une idée de la manière dont vous pouvez facilement visualiser les données de votre graphe à l'aide de graph-explorer :



Logiciel Tom Sawyer

Tom Sawyer Perspectives est une plateforme de développement dédiée à la visualisation et à l'analyse de données et de graphe à faible code pour les données stockées dans Amazon Neptune. Des interfaces de conception et de prévisualisation intégrées ainsi que de vastes bibliothèques d'API vous permettent de créer rapidement des applications de visualisation personnalisées de qualité professionnelle. Grâce à une interface de conception de type « pointer-cliquer » et à 30 algorithmes d'analyse intégrés, vous pouvez concevoir et développer des applications pour mieux comprendre les données fédérées à partir de dizaines de sources.

Le navigateur de base de données orientée graphe Tom Sawyer facilite la visualisation et l'analyse des données dans Amazon Neptune. Vous pouvez voir et comprendre les connexions entre vos données sans avoir une connaissance approfondie du langage ou du schéma de requête. Vous pouvez interagir avec les données sans connaissances techniques en chargeant simplement les voisins des nœuds sélectionnés et en élaborant la visualisation dans la direction souhaitée. Vous pouvez également tirer parti de cinq mises en page uniques pour afficher le graphe de la manière la plus significative possible, et vous pouvez appliquer des analyses de centralité, de regroupement et de recherche de chemin pour révéler des modèles inédits. Pour voir un exemple d'intégration du navigateur de base de données orientée graphe dans Neptune, consultez ce billet de blog. Pour

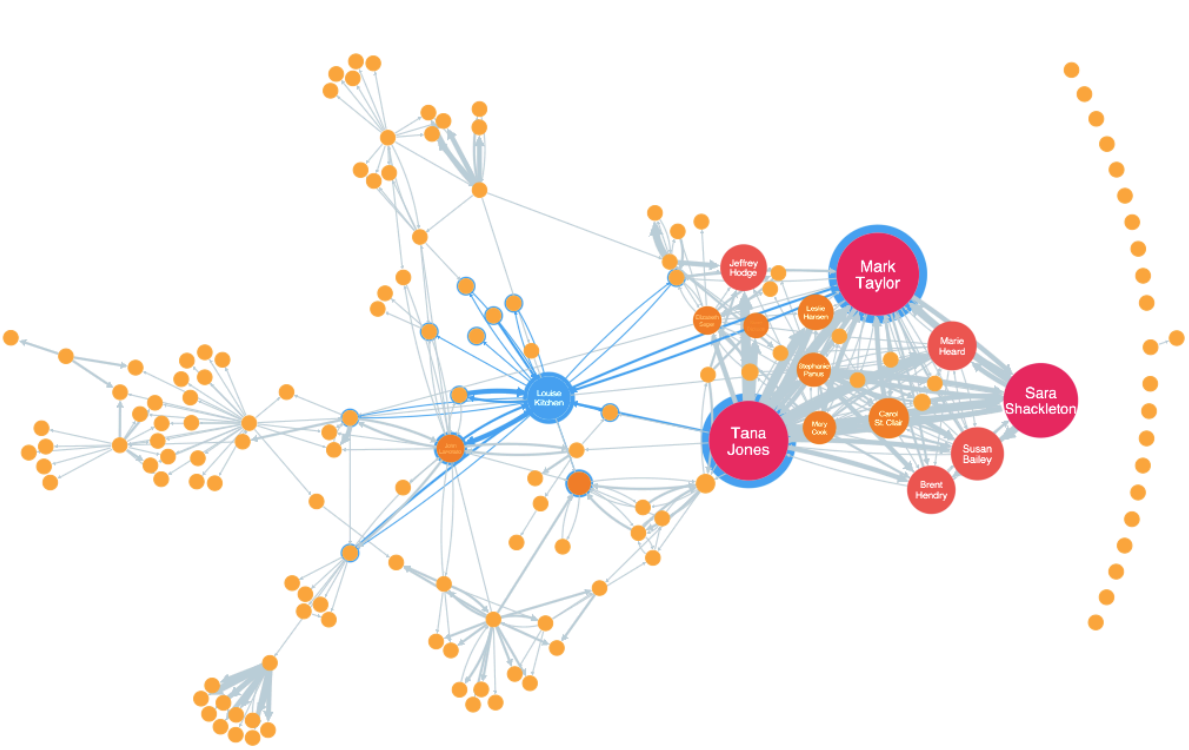
profiter d'une version d'évaluation gratuite du navigateur de base de données orientée graphe, rendez-vous sur AWS Marketplace.

Cambridge Intelligence

Cambridge Intelligence fournit des technologies de visualisation des données pour explorer et comprendre les données Amazon Neptune. Les boîtes à outils de visualisation de graphes (KeyLinespour JavaScript les développeurs et ReGraphpour les développeurs React) permettent de créer facilement des outils hautement interactifs et personnalisables pour les applications Web. Ces boîtes à outils exploitent WebGL et HTML5 Canvas pour des performances rapides. Elles prennent en charge des fonctions d'analyse de graphe avancées et associent flexibilité et capacité de mise à l'échelle à une architecture sécurisée et robuste. Ces kits SDK fonctionnent à la fois avec Neptune Gremlin et avec les données RDF.

Consultez ces didacticiels d'intégration pour les données Gremlin, les données SPARQL et l'architecture Neptune.

Voici un exemple de KeyLines visualisation :

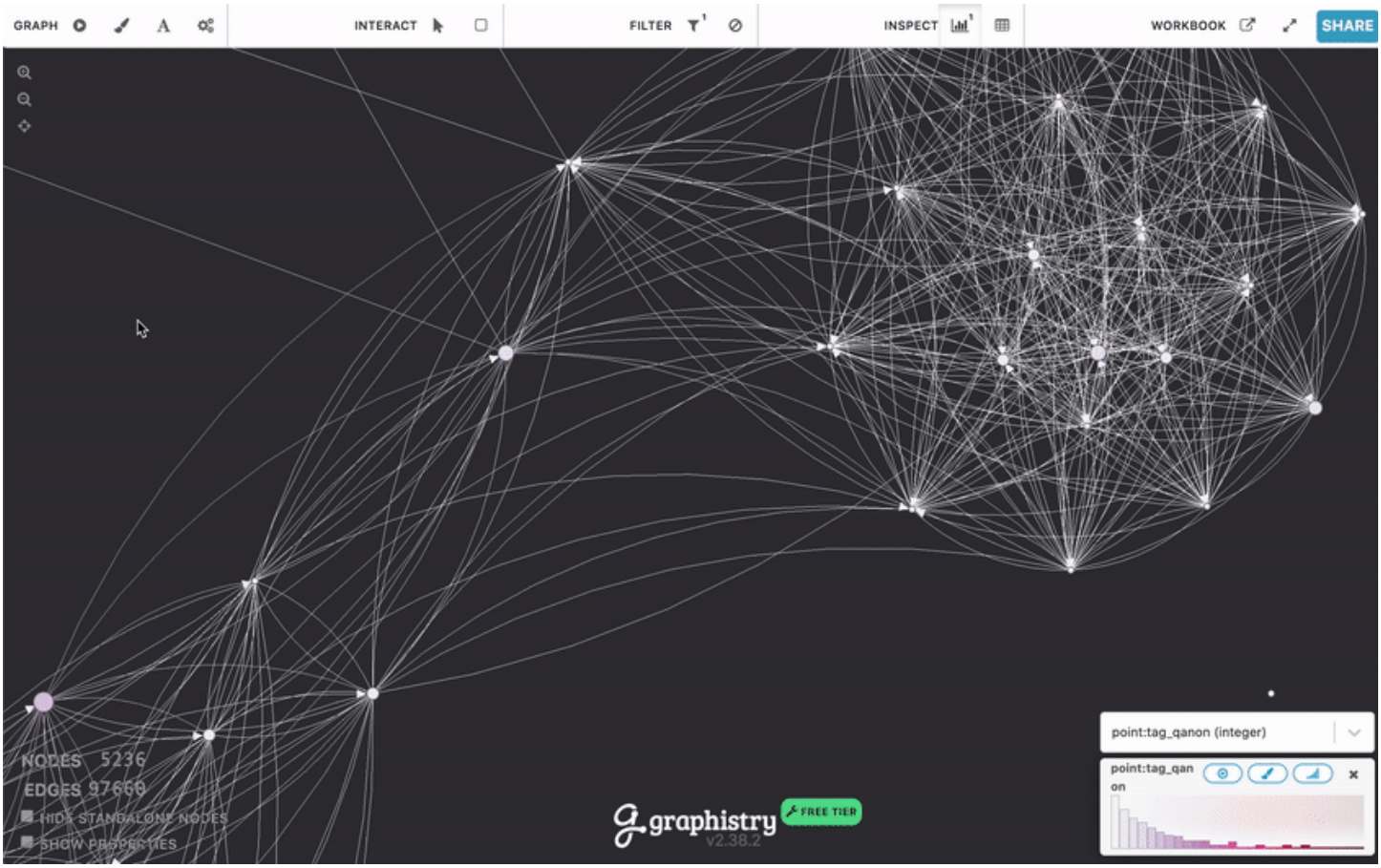


Graphistry

Graphistry est une plateforme d'intelligence visuelle orientée graphe qui tire parti de l'accélération du GPU pour créer des expériences visuelles riches. Les équipes peuvent collaborer sur Graphistry à l'aide de diverses fonctionnalités, allant de l'exploration sans code de fichiers et de bases de données, au partage de blocs-notes Jupyter et de tableaux de bord Streamlit, en passant par l'utilisation de l'API d'intégration dans vos propres applications.

Pour faire vos premiers pas avec des tableaux de bord entièrement interactifs à faible codage, configurez et lancez simplement le graph-app-kit et modifez seulement quelques lignes de code. Consultez ce billet de blog pour découvrir comment créer votre premier tableau de bord à l'aide de Graphistry et Neptune. Vous pouvez également essayer la démo de Neptune PyGraphistry. PyGraphistry est une bibliothèque d'analyse graphique visuelle en Python pour ordinateurs portables. Consultez ce carnet de didacticiels pour une démonstration de Neptune PyGraphistry .

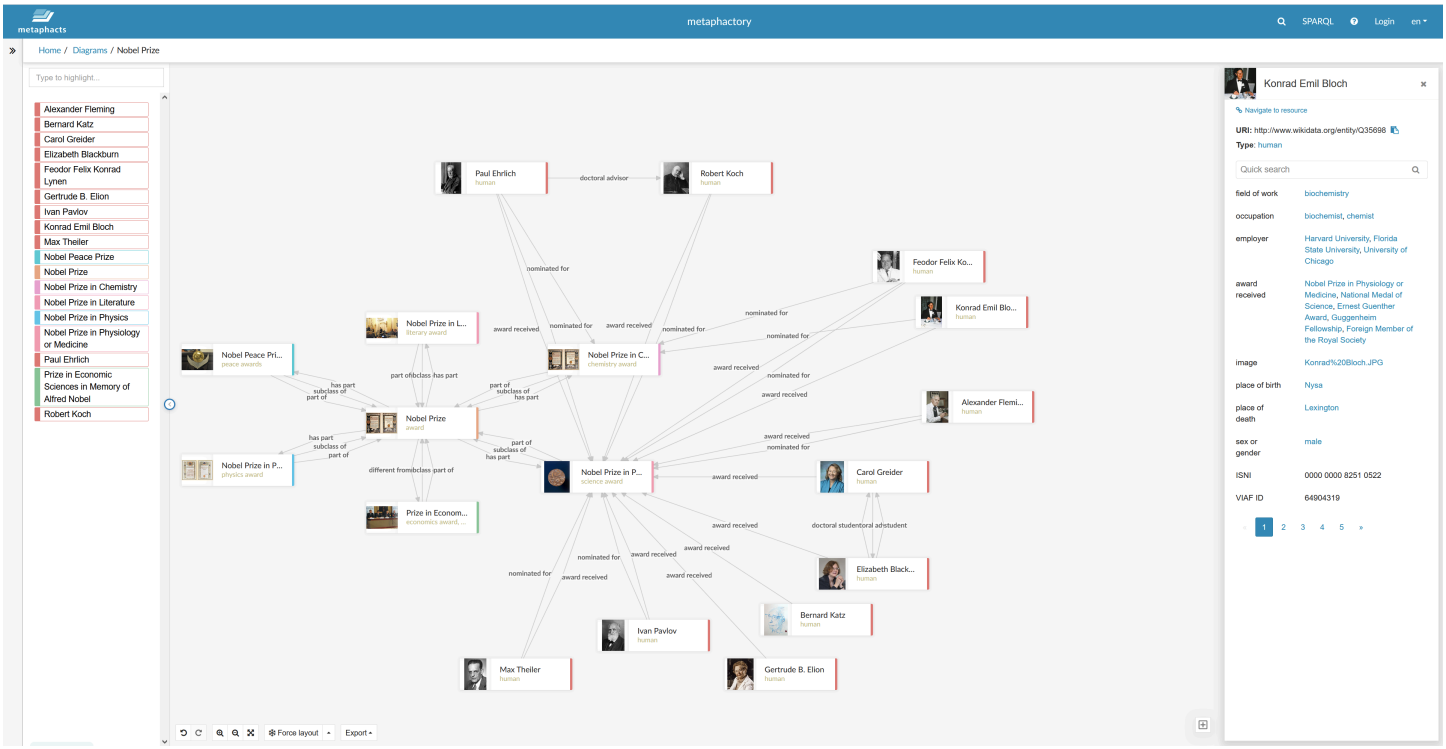
Pour commencer, rendez-vous sur Graphistry in the AWS Marketplace.



metaphacts

metaphacts propose une plateforme ouverte et flexible pour décrire et interroger des données de graphes, ainsi que pour visualiser et interagir avec des graphes de connaissances. À l'aide de metaphactory, vous pouvez créer des applications web interactives telles que des visualisations et des tableaux de bord sur des graphes de connaissances dans Neptune à l'aide du modèle de données RDF. La plateforme metaphactory permet une expérience de développement à faible code grâce à une interface utilisateur permettant le chargement des données, une interface visuelle de modélisation d'ontologie compatible OWL et SHACL, une interface utilisateur de requêtes et un catalogue de requêtes SPARQL, ainsi qu'un ensemble complet de composants web pour l'exploration, la visualisation, la recherche et la création de graphes.

Voici un exemple de visualisation de metaphactory :



Cette plateforme est destinée aux domaines de l'ingénierie, de la fabrication, de l'industrie pharmaceutique, des sciences de la vie, de la finance, de l'assurance, etc. Pour découvrir un exemple d'architecture de solution, consultez ce billet de blog.

Pour profiter d'une version d'évaluation gratuite de metaphactory, rendez-vous sur AWS Marketplace.

G.V() client de base de données de graphes

G.V() est un client de base de données graphique tout-en-un conçu pour permettre aux développeurs et aux professionnels des données d'explorer et d'interagir avec les données graphiques d'Amazon Neptune.

Avec G.V (), vous pouvez accélérer vos projets de graphes et effectuer les tâches quotidiennes de base de données de graphes de manière plus efficace. Elle est disponible via AWS Marketplace ou en tant qu'application de bureau autonome pour Windows, macOS et Linux.

Fonctionnalités principales

Éditeurs de requêtes

Les éditeurs de requêtes intégrés Gremlin et Cypher fournissent une saisie semi-automatique en temps réel, des suggestions et de la documentation lorsque vous tapez votre requête. Ces

fonctionnalités s'appuient sur le schéma de données de votre graphe pour vous aider à rédiger rapidement des requêtes précises.

Vous pouvez paramétrer et enregistrer les requêtes sous forme de rapports en un clic, ce qui vous permet de créer des rapports pertinents et centralisés pour votre base de données de graphes en quelques minutes.

Formats de visualisation des données

Pour analyser les résultats de vos requêtes, G.V () propose plusieurs formats de visualisation des données :

- Visualisation graphique pour explorer les connexions et les modèles dans vos données, avec une sous-couche optionnelle pour les données cartographiques géographiques
- Vues tabulaires, organisation des sommets, des arêtes ou des résultats de requêtes tabulaires sous forme de lignes pour une comparaison rapide des valeurs de propriétés
- Format de données JSON pour une sortie conviviale pour les développeurs
- Navigateur d'objets pour naviguer dans vos données en fonction de sa structure hiérarchique, idéal pour les agrégations complexes d'informations

Explorateur de données graphiques

G.VL'explorateur de données graphiques sans code de () vous permet de rechercher facilement les données de votre graphe à l'aide de filtres de propriétés et de parcourir les arêtes depuis les sommets de manière interactive via la vue graphique. Vous pouvez également créer, mettre à jour et supprimer des sommets, des arêtes et leurs propriétés en quelques clics pour modifier les données de votre graphe sans avoir à écrire de requêtes complètes.

Vue du modèle de données graphiques

La vue du modèle de données graphiques fournit une représentation visuelle détaillée de votre schéma de graphe de propriétés étiquetées afin d'aider votre équipe à comprendre votre modèle de données.

Intégration avec Amazon Neptune

G.V() propose une intégration approfondie avec Amazon Neptune, notamment :

- Authentification IAM
- Support de l'API Neptune
- Analyse lente des requêtes
- Informations sur le journal d'audit

Consultez la documentation G.V () pour en savoir plus.

Disponibilité et mises à jour

G.V() est en constante évolution avec de nouvelles fonctionnalités publiées chaque mois. Commencez dès aujourd'hui par un essai gratuit de la version de bureau ou de la version AWS Marketplace.

Démonstration du produit

Voir G.V () en action :

En savoir plus

Pour plus d'informations sur les fonctionnalités et les capacités de G.V (), consultez la documentation de G.V ().

Linkurious

Linkurious fournit aux utilisateurs techniques et non techniques une plateforme d'intelligence décisionnelle contextuelle de nouvelle génération, adaptée aux entreprises, alimentée par une technologie graphique native.

Utilisé par les agences du secteur public, les entreprises du classement Global 2000 et les organisations à but non lucratif, Linkurious aide les chercheurs, les analystes et les enquêteurs axés sur les données à détecter des modèles cachés dans des données connectées complexes et à prendre des décisions plus rapides en fonction du contexte.

Pour les utilisateurs de Neptune, Linkurious propose plusieurs solutions logicielles avancées d'analyse graphique et de visualisation permettant aux entreprises de visualiser, d'explorer et d'analyser les données connectées stockées dans la base de données Neptune.

Entreprise Linkurious

Linkurious Enterprise est un module autonome de la plateforme Linkurious Decision Intelligence. Il s'agit d'un logiciel de visualisation et d'analyse de graphes prêt à l'emploi qui aide les utilisateurs à exploiter de manière visuelle et intuitive les données stockées dans Neptune. Grâce à son interface conviviale, à ses capacités d'analyse avancées et à ses fonctionnalités de sécurité de niveau professionnel, Linkurious Enterprise permet aux équipes de détecter des modèles complexes, de découvrir des connexions cachées et d'accélérer les décisions basées sur les données du graphe Neptune.

- Visualisez les données de Neptune : connectez-vous facilement à Neptune et accédez à une vue à 360° de vos données graphiques pour une exploration dynamique et interactive.
- Accélérez l'analyse des données : filtrez les résultats, explorez les chronologies, passez en mode géographique ou utilisez le générateur de requêtes sans code et prédéfinissez des requêtes spécifiques pour tous les utilisateurs afin de découvrir rapidement des informations pertinentes.
- Détectez automatiquement les modèles complexes : utilisez des alertes multimodèles basées sur des graphes et la déduplication des alertes pour faire apparaître automatiquement les modèles intéressants dans le graphique et réduire les efforts de triage.
- Collaborez efficacement : partagez des vues graphiques et des résultats d'investigation entre les équipes ou au niveau de l'organisation pour favoriser une prise de décision cohérente et basée sur les données.
- Déploiement sécurisé en entreprise : garantissez la conformité grâce au contrôle d'accès basé sur les rôles, à LDAP/SSO l'intégration et à la journalisation des audits.

Pour obtenir des informations techniques, consultez la documentation de Linkurious Enterprise.

Vous souhaitez tester Linkurious Enterprise ? Demandez un essai gratuit de 30 jours.

Consultez les tarifs et les options d'abonnement sur AWS Marketplace.

Ogma

Ogma est une JavaScript bibliothèque commerciale développée par Linkurious, conçue pour les développeurs qui créent des visualisations graphiques interactives puissantes et à grande échelle dans des applications exploitant les données Neptune.

Voici les principales fonctionnalités :

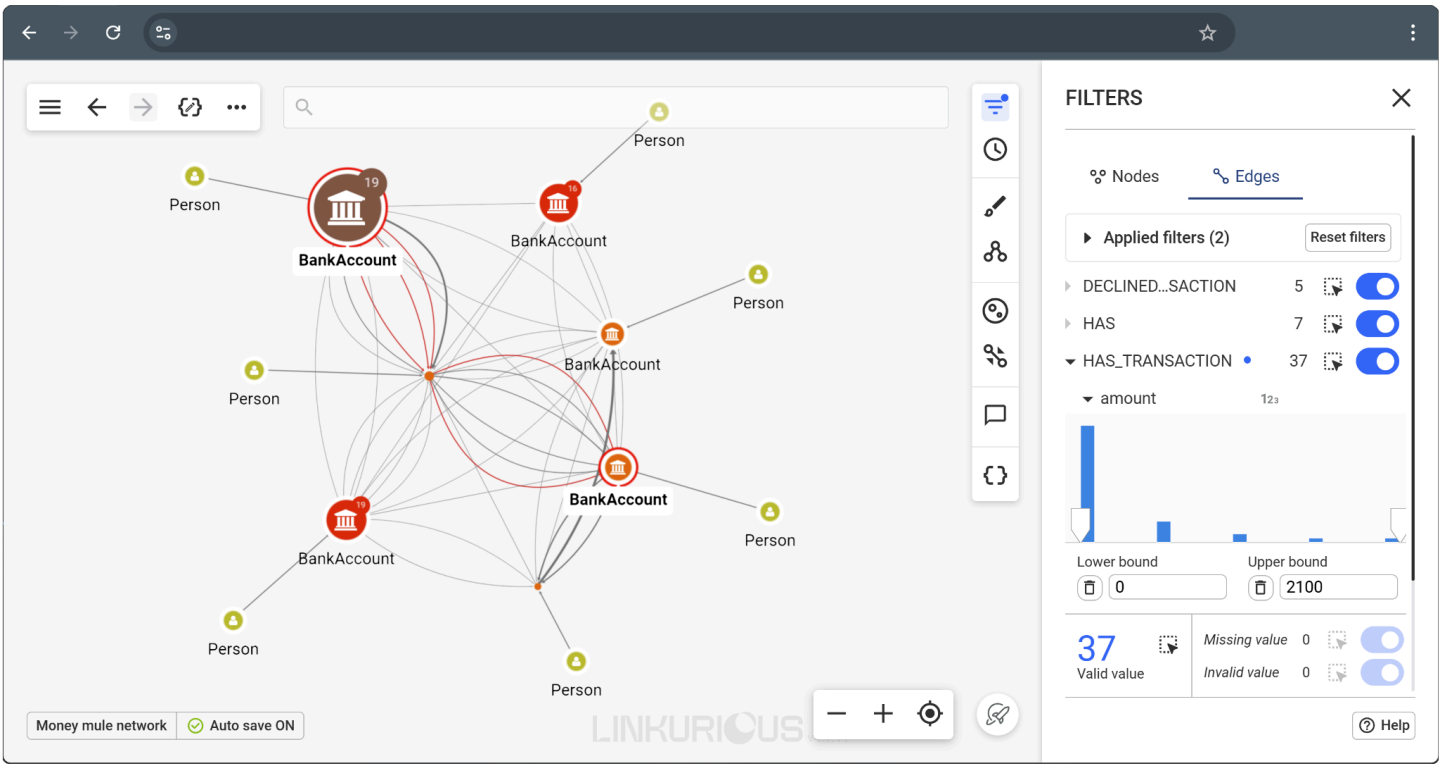
- High-performance rendu : utilisez WebGL et des algorithmes propriétaires pour visualiser des dizaines de milliers de nœuds presque instantanément tout en préservant la précision et la qualité, même avec un GPU vieux de 5 ans, afin de garantir une expérience utilisateur fluide.
- Personnalisation : personnalisez les visualisations avec une variété de mises en page, de styles et de fonctionnalités interactives. Enrichissez les visualisations avec des algorithmes graphiques intégrés ou des classificateurs d'apprentissage automatique externes. Ajoutez des superpositions visuelles et des annotations intégrées à l'image pour obtenir des informations et une collaboration humaines.
- Intégration : commencez rapidement grâce à une documentation complète, à des didacticiels et à un terrain de jeu de code. Chargez et transformez les données en quelques minutes, intégrez Ogma dans votre framework et accédez à une assistance avancée pour les développeurs.

Pour des informations plus détaillées et des didacticiels sur l'intégration d'Ogma à Neptune, consultez la documentation d'Ogma.

Intéressé à essayer Ogma ? Demandez un essai gratuit de 30 jours.

Consultez les tarifs et les options d'abonnement sur AWS Marketplace.

Voici une visualisation de Linkurious Enterprise :



Graph.Build

Dans tous les domaines, la collaboration avec des experts en la matière est essentielle pour concevoir des modèles de graphes qui répondent efficacement à des cas d'utilisation spécifiques. Que vous abordiez des charges de travail classiques telles que l'analyse de graphes ou que vous exploriez des applications d'intelligence artificielle avancées telles que GraphRag, la prédiction de liens ou le raisonnement neurosymbolique, la capacité de créer et d'itérer sur des modèles de graphes est essentielle. La conception et le perfectionnement des modèles graphiques deviennent ainsi plus accessibles, ce qui permet aux experts du domaine de travailler main dans la main avec les ingénieurs et les analystes de données pour créer des solutions personnalisées percutantes. Graph.Build

La Graph.Buildplateforme est un studio de conception de modèles graphiques et une usine de construction dédiés et sans code.

Graph.Build est disponible sur le AWS Marketplace.

Graph.Build Atelier :

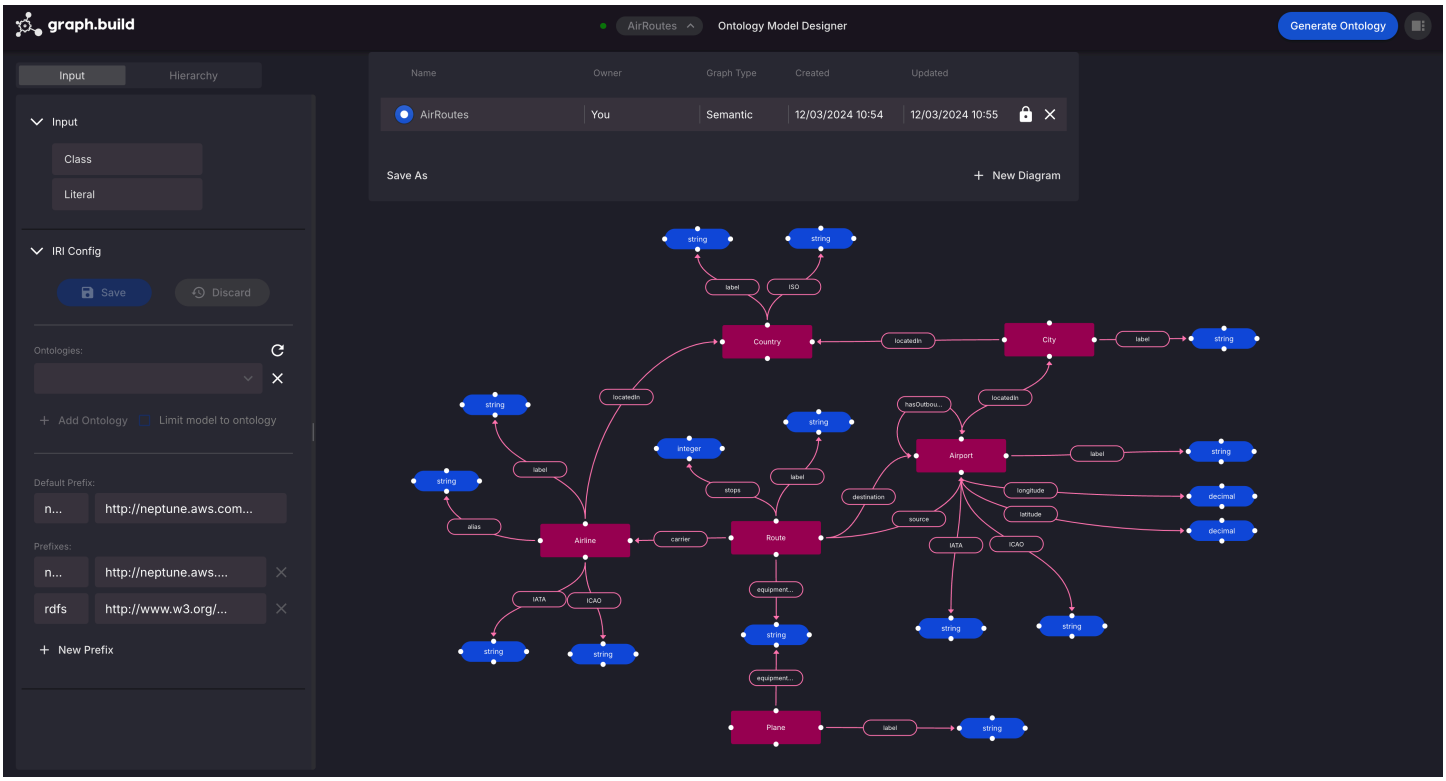
Studio est un IDE de conception graphique collaboratif sans code. Vous pouvez utiliser le studio pour concevoir des schémas pour des graphes GPL, des ontologies pour des graphes RDF, des modèles ETL, des configurations de capture de données modifiées et même des points de terminaison de graphes SPARQL virtuels.

Graph.Build Transformateurs :

Les transformateurs créent des modèles de graphes à partir de données réelles. Les modèles de graphes de transformateurs peuvent être conçus dans le studio, ce qui signifie que la conception, la création et le test de modèles de graphes ne nécessitent aucun code. Vous pouvez connecter des transformateurs à un large éventail de sources de données, notamment Amazon Aurora, Amazon Athena, Amazon S3 et Amazon MSK. Les transformateurs disposent d'un large éventail de fonctionnalités permettant de gérer les modèles de graphes présents dans d'autres ensembles de données, notamment la capture des données de modification (CDC), la virtualisation et la matérialisation en mode mise à jour ou insertion.

Graph.Build Écrivains :

Les rédacteurs sont configurés pour écrire ou mettre à jour des modèles de graphes Labelled Property Graph (LPG) et Resource Description Framework (RDF) dans des bases de données de graphes cibles, comme Amazon Neptune et Neptune Analytics.



Demandez une démonstration dès aujourd'hui.

Exportation de données depuis un cluster de bases de données Neptune

Il existe plusieurs méthodes efficaces pour exporter des données depuis un cluster de bases de données Neptune :

- Pour de petites quantités de données, utilisez simplement les résultats d'une ou de plusieurs requêtes.
- Pour les données RDF, le protocole GSP (Graph Store Protocol) peut faciliter l'exportation. Par exemple :

```
curl --request GET \
  'https://your-neptune-endpoint:port/sparql/gsp/?graph=http://www.example.com/named/graph'
```

- Il existe également un outil open source puissant et flexible permettant d'exporter les données Neptune, à savoir `neptune-export`. Les sections suivantes décrivent les fonctionnalités de cet outil et expliquent comment les utiliser.

Rubriques

- Utiliser neptune-export
- Utilisation du Neptune-Export service pour exporter les données Neptune
- Utilisation de l'outil de ligne de commande neptune-export pour exporter des données depuis Neptune
- Fichiers exportés par Neptune-Export et neptune-export
- Paramètres utilisés pour contrôler le processus d'exportation de Neptune
- Résolution des problèmes liés au processus d'exportation de Neptune
- Exportation des résultats de requêtes Gremlin vers Amazon S3

Utiliser **`neptune-export`**

Vous pouvez utiliser l'outil `neptune-export` open source de deux manières différentes :

- En tant que service d'exportation Neptune. Lorsque vous exportez des données depuis Neptune à l'aide du service d'exportation Neptune, vous déclenchez et surveillez les tâches d'exportation via une API REST.
- En tant qu'utilitaire de ligne de commande Java **`neptune-export`**. Pour utiliser cet outil de ligne de commande en vue d'exporter des données Neptune, vous devez l'exécuter dans un environnement dans lequel votre cluster de bases de données Neptune est accessible.

Le service d'exportation Neptune et l'outil de ligne de commande `neptune-export` publient les données dans Amazon Simple Storage Service (Amazon S3). Les données sont chiffrées à l'aide du chiffrement côté serveur Amazon S3 (`SSE-S3`).

> **Note**
>
> Il est recommandé d'activer la journalisation des accès sur tous les compartiments Amazon S3, afin de pouvoir auditer tous les accès à ces compartiments.

Si vous essayez d'exporter des données depuis un cluster de bases de données Neptune dont les données sont modifiées pendant l'exportation, la cohérence des données exportées n'est pas garantie. En d'autres termes, si le cluster gère le trafic d'écriture alors qu'une tâche d'exportation est en cours, les données exportées peuvent présenter des incohérences. Cela est possible que vous exportiez les données à partir de l'instance principale du cluster ou à partir d'un ou de plusieurs réplicas en lecture.

Pour garantir la cohérence des données exportées, il est préférable de les exporter à partir d'un clone du cluster de bases de données. Cette approche fournit à l'outil d'exportation une version statique de vos données et garantit que la tâche d'exportation ne ralentit pas les requêtes dans le cluster de bases de données d'origine.

Pour faciliter les choses, vous pouvez indiquer que vous souhaitez cloner le cluster de bases de données source lorsque vous déclenchez une tâche d'exportation. Dans ce cas, le processus d'exportation crée automatiquement le clone, l'utilise pour l'exportation, puis le supprime une fois l'exportation terminée.

Utilisation du Neptune-Export service pour exporter les données Neptune

Vous pouvez suivre les étapes suivantes pour exporter les données de votre cluster de base de données Neptune vers Amazon S3 à l'aide du Neptune-Export service :

Installation du Neptune-Export service

Utilisez un AWS CloudFormation modèle pour créer la pile :

Pour installer le Neptune-Export service

1. Lancez la CloudFormation pile sur la CloudFormation console en cliquant sur l'un des boutons Launch Stack dans le tableau suivant :

Région	Afficher	Afficher dans Designer	Lancer
USA Est (Virginie du Nord)	Afficher	Afficher dans Designer	
USA Est (Ohio)	Afficher	Afficher dans Designer	
USA Ouest (Californie du Nord)	Afficher	Afficher dans Designer	
USA Ouest (Oregon)	Afficher	Afficher dans Designer	
Canada (Centre)	Afficher	Afficher dans Designer	
Amérique du Sud (São Paulo)	Afficher	Afficher dans Designer	
Europe (Stockholm)	Afficher	Afficher dans Designer	

Région	Afficher	Afficher dans Designer	Lancer
Europe (Irlande)	Afficher	Afficher dans Designer	
Europe (Londres)	Afficher	Afficher dans Designer	
Europe (Paris)	Afficher	Afficher dans Designer	
Europe (Francfort)	Afficher	Afficher dans Designer	
Middle East (Bahrain)	Afficher	Afficher dans Designer	
Moyen-Orient (EAU)	Afficher	Afficher dans Designer	
Israël (Tel Aviv)	Afficher	Afficher dans Designer	
Afrique (Le Cap)	Afficher	Afficher dans Designer	
Asie-Pacifique (Hong Kong)	Afficher	Afficher dans Designer	
Asie-Pacifique (Tokyo)	Afficher	Afficher dans Designer	
Asie-Pacifique (Séoul)	Afficher	Afficher dans Designer	
Asie-Pacifique (Singapour)	Afficher	Afficher dans Designer	

Région	Afficher	Afficher dans Designer	Lancer
Asie-Pacifique (Sydney)	Afficher	Afficher dans Designer	
Asie-Pacifique (Mumbai)	Afficher	Afficher dans Designer	
Chine (Pékin)	Afficher	Afficher dans Designer	
Chine (Ningxia)	Afficher	Afficher dans Designer	
AWS GovCloud (US-West)	Afficher	Afficher dans Designer	
AWS GovCloud (US-East)	Afficher	Afficher dans Designer	

2. Sur la page Select Template, choisissez Next.
3. Sur la page Spécifier les détails, vérifiez les paramètres suivants :

   - **`VPC`**— Le moyen le plus simple de configurer le Neptune-Export service est de l'installer dans le même Amazon VPC que votre base de données Neptune. Si vous souhaitez l'installer dans un VPC distinct, vous pouvez utiliser le peering VPC pour établir la connectivité entre le VPC du cluster de base de données Neptune et le VPC de service. Neptune-Export
   - **`Subnet1`**— Le Neptune-Export service doit être installé dans un sous-réseau de votre VPC qui autorise le trafic HTTPS IPv4 sortant du sous-réseau vers Internet. Cela permet au Neptune-Export service d'appeler l'API AWS Batch pour créer et exécuter une tâche d'exportation.

     Si vous avez créé votre cluster Neptune à l'aide du CloudFormation modèle figurant sur la Créer un cluster Neptune page de la documentation de Neptune, vous pouvez utiliser les `PrivateSubnet2` sorties `PrivateSubnet1` et de cette pile pour renseigner ce paramètre et le suivant.

- **`Subnet2`** : deuxième sous-réseau du VPC qui autorise le trafic HTTPS IPv4 sortant du sous-réseau vers Internet.
- **`EnableIAM`**— Définissez ce paramètre sur `true` pour sécuriser l' Neptune-Endpoint API à l'aide de Gestion des identités et des accès AWS (IAM). Nous vous recommandons de le faire.

  Si vous activez l'authentification IAM, vous devez signer avec `Sigv4` toutes les demandes HTTPS envoyées au point de terminaison. Vous pouvez utiliser un outil comme awscurl pour signer les demandes en votre nom.
- **`VPCOnly`**— En le définissant comme `true` point de terminaison d'exportation VPC-only, vous ne pouvez y accéder que depuis le VPC sur lequel le Neptune-Export service est installé. Cela limite l'utilisation de l' Neptune-Export API uniquement depuis ce VPC.

  Nous vous recommandons de définir le `VPCOnly` à la valeur `true`.
- **`NumOfFilesULimit`** : spécifiez une valeur comprise entre 10 000 et 1 000 000 pour `nofile` dans la propriété du conteneur `ulimits`. La valeur par défaut est 10 000, et nous vous recommandons de la conserver, sauf si le graphe contient un grand nombre d'étiquettes uniques.
- **`PrivateDnsEnabled`** (valeur booléenne) : indique si vous souhaitez associer une zone hébergée privée au VPC spécifié. La valeur par défaut est `true`.

  Lorsqu'un point de terminaison VPC est créé et que cet indicateur est activé, tout le trafic API Gateway est acheminé via le point de terminaison VPC, et les appels de point de terminaison API Gateway publics sont désactivés. Si vous définissez `PrivateDnsEnabled` sur `false`, le point de terminaison API Gateway public est activé, mais le service d'exportation Neptune ne peut pas être connecté via le point de terminaison DNS privé. Vous pouvez ainsi utiliser un point de terminaison DNS public pour que le point de terminaison VPC appelle le service d'exportation, comme indiqué ici.
- **`NeptuneExportVersion`**— Spécifiez la version de l'utilitaire d'exportation Neptune à utiliser. Toutes les versions supérieures ou égales à `v1.1.11` sont prises en charge. Une version de `v2.latest` peut être utilisée pour recevoir automatiquement des mises à jour mineures. La liste complète des versions disponibles, ainsi que les notes de mise à jour, se trouvent dans les GitHub versions open source.

4. Choisissez Suivant.
5. Dans la page Options, choisissez Suivant.

6. Sur la page Vérification, cochez la première case pour accepter que CloudFormation créera des ressources IAM. Cochez la deuxième case pour confirmer `CAPABILITY_AUTO_EXPAND` comme nouvelle pile.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` accepte explicitement que les macros soient étendues lors de la création de la pile, sans révision préalable. Les utilisateurs créent souvent un jeu de modifications à partir d'un modèle traité, de sorte que les modifications apportées par les macros puissent être révisées avant la création de la pile. Pour en savoir plus, consultez l'API CloudFormation CreateStack.

   Ensuite, choisissez Créer.

Activez l'accès à Neptune depuis Neptune-Export

Une fois l' Neptune-Export installation terminée, mettez à jour votre groupe de sécurité Neptune VPC pour autoriser l'accès depuis. Neptune-Export Lorsque la Neptune-Export CloudFormation pile a été créée, l'onglet Sorties inclut un `NeptuneExportSecurityGroup` ID. Mettez à jour votre groupe de sécurité Neptune VPC pour autoriser l'accès depuis ce groupe de sécurité. Neptune-Export

Activer l'accès au Neptune-Export point de terminaison depuis une VPC-based instance EC2

Si vous créez votre Neptune-Export point de terminaison VPC-only, vous ne pouvez y accéder que depuis le VPC dans lequel le Neptune-Export service est installé. Pour autoriser la connectivité depuis une instance Amazon EC2 dans le VPC à partir de laquelle vous pouvez effectuer des appels d' Neptune-Export API, attachez l'instance `NeptuneExportSecurityGroup` créée par la CloudFormation pile à cette instance Amazon EC2.

Exécuter une Neptune-Export tâche à l'aide de l' Neptune-Export API

L'onglet Sorties de la CloudFormation pile inclut également le`NeptuneExportApiUri`. Utilisez cette URI chaque fois que vous envoyez une demande au Neptune-Export point de terminaison.

Exécuter une tâche d'exportation

- Assurez-vous que l'utilisateur ou le rôle sous lequel l'exportation est exécutée dispose d'une autorisation `execute-api:Invoke`.
- Si vous définissez le `EnableIAM` paramètre sur « `true` in the CloudFormation stack » lors de l'installation Neptune-Export, vous devez `Sigv4` signer toutes les demandes à l' Neptune-Export API. Nous vous recommandons d'utiliser awscurl pour envoyer des demandes à l'API. Tous les exemples présentés ici supposent que l'authentification IAM est activée.
- Si vous définissez le `VPCOnly` paramètre sur `true` in the CloudFormation stack lors de l'installation Neptune-Export, vous devez appeler l' Neptune-Export API depuis le VPC, généralement depuis une instance Amazon EC2 située dans le VPC.

Pour commencer à exporter des données, envoyez une demande au point de terminaison `NeptuneExportApiUri` avec des paramètres de demande `command` et `outputS3Path`, ainsi qu'un paramètre d'exportation `endpoint`.

Le message suivant est un exemple de demande exportant les données du graphe de propriétés de Neptune et les publiant sur Amazon S3 :

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-pg",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": { "endpoint": "(your Neptune endpoint DNS name)" }
      }'
```

De même, voici un exemple de demande qui exporte des données RDF de Neptune vers Amazon S3 :

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-rdf",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
```

```
        "params": { "endpoint": "(your Neptune endpoint DNS name)" }
      }'
```

Si vous omettez le paramètre de `command` requête, Neptune-Export tente par défaut d'exporter les données du graphe de propriétés depuis Neptune.

Si la commande précédente a été exécutée avec succès, le résultat ressemblera à ceci :

```
{
  "jobName": "neptune-export-abc12345-1589808577790",
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f"
}
```

Surveillance de la tâche d'exportation que vous venez de commencer

Pour surveiller une tâche en cours d'exécution, ajoutez son JobID à l'élément `NeptuneExportApiUri`, comme ceci :

```
curl \
  (your NeptuneExportApiUri)/(the job ID)
```

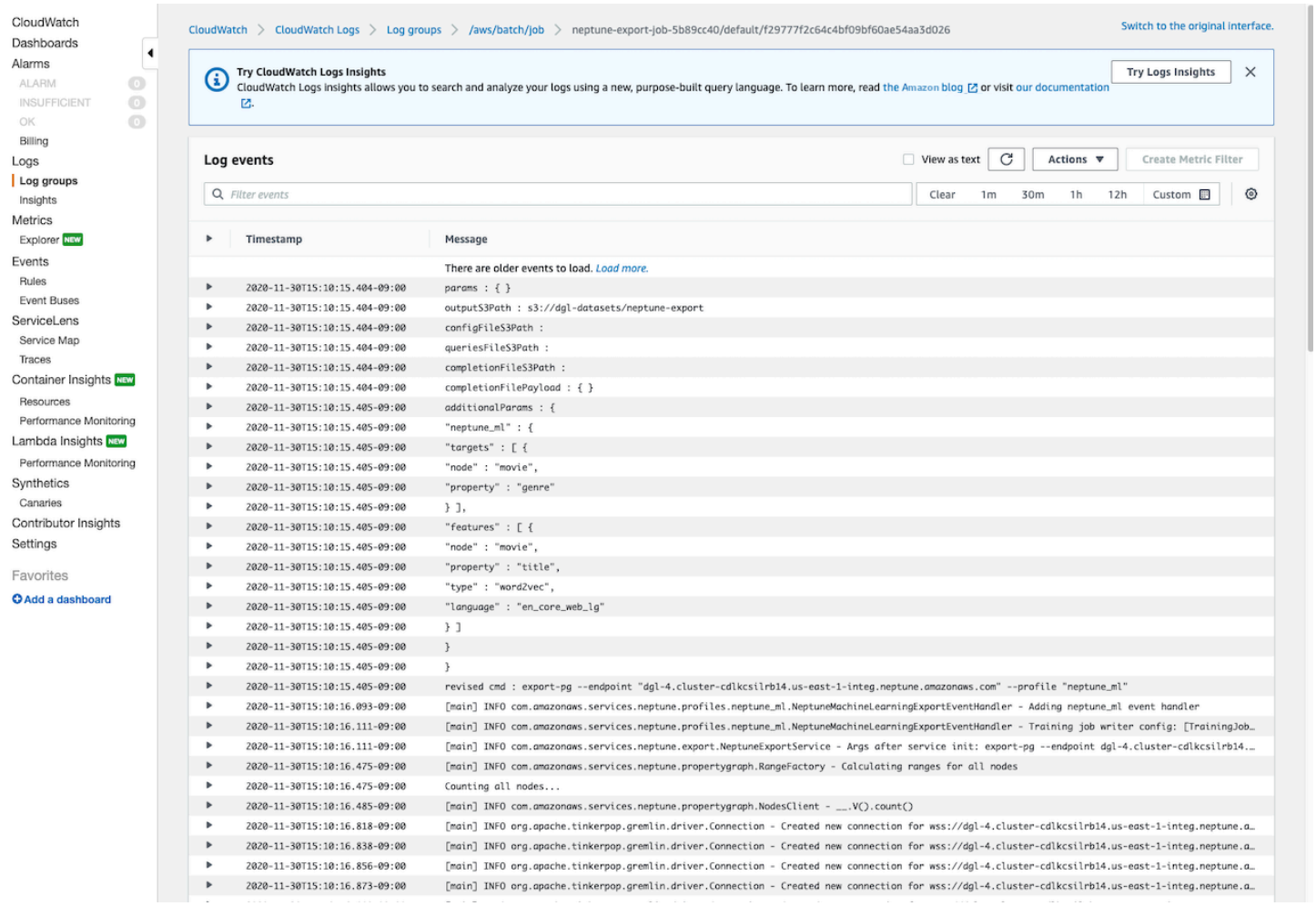
Si le service n'a pas encore lancé la tâche d'exportation, la réponse est la suivante :

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "pending"
}
```

Lorsque vous répétez cette commande après le début de la tâche d'exportation, la réponse se présente comme suit :

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "running",
  "logs": "https://us-east-1.console.aws.amazon.com/cloudwatch/home?..."
}
```

Si vous ouvrez les CloudWatch journaux dans Logs à l'aide de l'URI fourni par l'appel d'état, vous pouvez ensuite suivre la progression de l'exportation en détail :



Annulation d'une tâche d'exportation en cours d'exécution

Pour annuler une tâche d'exportation en cours à l'aide du AWS Management Console

1. Ouvrez la AWS Batch console à l'adresse https://console.aws.amazon.com/batch/.
2. Choisissez Tâches.
3. Localisez la tâche en cours que vous souhaitez annuler, en fonction de son `jobID`.
4. Sélectionnez Annuler la tâche.

Pour annuler une tâche d'exportation en cours à l'aide de l'API d'exportation Neptune :

Envoyez une demande `HTTP DELETE` à `NeptuneExportApiUri` en y ajoutant `jobID`, comme ceci :

```
curl -X DELETE \
```

```
(your NeptuneExportApiUri)/(the job ID)
```

Utilisation de l'outil de ligne de commande **neptune-export** pour exporter des données depuis Neptune

Vous pouvez suivre les étapes suivantes pour exporter les données du cluster de bases de données Neptune vers Amazon S3 à l'aide de l'utilitaire de ligne de commande `neptune-export` :

Conditions préalables à l'utilisation de l'utilitaire de ligne de commande **neptune-export**

Avant de commencer

- Disposer de la version 8 du kit JDK : la version 8 du kit de développement Java SE (JDK) doit être installée.
- Téléchargez l'utilitaire neptune-export : téléchargez et installez le fichier neptune-export.jar.
- Assurez-vous que **neptune-export** a accès à votre VPC Neptune : exécutez neptune-export depuis un emplacement permettant d'accéder au VPC où se trouve votre cluster de bases de données Neptune.

  Par exemple, vous pouvez l'exécuter sur une instance Amazon EC2 au sein du VPC Neptune, dans un VPC distinct associé au VPC Neptune ou sur un hôte bastion distinct.
- Assurez-vous que les groupes de sécurité VPC accordent l'accès à **neptune-export** : vérifiez que le ou les groupes de sécurité VPC attachés au VPC Neptune autorisent l'accès au cluster de bases de données à partir de l'adresse IP ou du groupe de sécurité associé à l'environnement `neptune-export`.
- Configurez les autorisations IAM nécessaires — Si l'authentification de base de données Gestion des identités et des accès AWS (IAM) est activée sur votre base de données, assurez-vous que le rôle sous lequel `neptune-export` s'exécute est associé à une politique IAM autorisant les connexions à Neptune. Pour en savoir plus sur les politiques Neptune, consultez Utilisation de politiques IAM.

  Si vous souhaitez utiliser le paramètre d'exportation `clusterId` dans vos demandes de requête, le rôle sous lequel `neptune-export` s'exécute nécessite les autorisations IAM suivantes :

  - `rds:DescribeDBClusters`

- `rds:DescribeDBInstances`
- `rds:ListTagsForResource`

Si vous souhaitez effectuer une exportation à partir d'un cluster cloné, le rôle sous lequel `neptune-export` s'exécute nécessite les autorisations IAM suivantes :

- `rds:AddTagsToResource`
- `rds:DescribeDBClusters`
- `rds:DescribeDBInstances`
- `rds:ListTagsForResource`
- `rds:DescribeDBClusterParameters`
- `rds:DescribeDBParameters`
- `rds:ModifyDBParameterGroup`
- `rds:ModifyDBClusterParameterGroup`
- `rds:RestoreDBClusterToPointInTime`
- `rds:DeleteDBInstance`
- `rds:DeleteDBClusterParameterGroup`
- `rds:DeleteDBParameterGroup`
- `rds:DeleteDBCluster`
- `rds:CreateDBInstance`
- `rds:CreateDBClusterParameterGroup`
- `rds:CreateDBParameterGroup`

Pour publier les données exportées sur Amazon S3, le rôle sous lequel `neptune-export` s'exécute nécessite les autorisations IAM suivantes pour le ou les sites Amazon S3 :

- `s3:PutObject`
- `s3:PutObjectTagging`
- `s3:GetObject`

- Définissez la variable d'environnement **`SERVICE_REGION`** : définissez la variable d'environnement `SERVICE_REGION` pour identifier la région dans laquelle se trouve votre cluster de bases de données (voir Connexion à Neptune pour obtenir une liste des identifiants de région).

Exécution de l'utilitaire **`neptune-export`** pour lancer une opération d'exportation

Utilisez la commande suivante pour exécuter neptune-export depuis la ligne de commande et démarrer une opération d'exportation :

```
java -jar neptune-export.jar nesvc \
  --root-path (path to a local directory) \
  --json (the JSON file that defines the export)
```

Cette commande utilise deux paramètres :

Paramètres de neptune-export lors du démarrage d'une exportation

- **`--root-path`** : chemin d'accès à un répertoire local dans lequel les fichiers d'exportation sont écrits avant d'être publiés sur Amazon S3.
- **`--json`** : objet JSON qui définit l'exportation.

Exemples de commandes impliquant l'utilitaire de ligne de commande **`neptune-export`**

Pour exporter les données du graphe de propriétés directement depuis votre cluster de bases de données source :

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
            }
          }'
```

Pour exporter des données RDF directement depuis votre cluster de bases de données source :

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
```

```
            "command": "export-rdf",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
            }
          }'
```

Si vous omettez le paramètre de demande `command`, l'utilitaire `neptune-export` exporte les données du graphe de propriétés depuis Neptune par défaut.

Pour effectuer une exportation depuis un clone de votre cluster de bases de données :

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)",
              "cloneCluster" : true
            }
          }'
```

Pour effectuer une exportation depuis votre cluster de bases de données à l'aide de l'authentification IAM :

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
              "useIamAuth" : true
            }
          }'
```

Fichiers exportés par Neptune-Export et **`neptune-export`**

Lorsqu'une exportation est terminée, les fichiers d'exportation sont publiés sur l'emplacement Amazon S3 que vous avez spécifié. Tous les fichiers publiés sur Amazon S3 sont chiffrés à l'aide

du chiffrement côté serveur Amazon S3 (`SSE-S3`). Les dossiers et fichiers publiés sur Amazon S3 varient selon que vous exportez des données du graphe de propriétés ou des données RDF. Si vous ouvrez l'emplacement Amazon S3 où les fichiers sont publiés, le contenu suivant s'affiche :

Emplacement des fichiers exportés dans Amazon S3

- **`nodes/`** : ce dossier contient des fichiers de données de nœuds au format CSV (valeurs séparées par des virgules) ou JSON.

  Dans Neptune, les nœuds peuvent avoir une ou plusieurs étiquettes. Les nœuds avec différentes étiquettes individuelles (ou différentes combinaisons de plusieurs étiquettes) sont écrits dans différents fichiers, ce qui signifie qu'aucun fichier individuel ne contient de données pour des nœuds dont les combinaisons d'étiquettes diffèrent. Si un nœud possède plusieurs étiquettes, celles-ci sont triées par ordre alphabétique avant d'être attribuées à un fichier.
- **`edges/`** : ce dossier contient des fichiers de données d'arêtes au format CSV (valeurs séparées par des virgules) ou JSON.

  Comme pour les fichiers de nœuds, les données d'arêtes sont écrites dans différents fichiers en fonction d'une combinaison de leurs étiquettes. À des fins d'entraînement du modèle, les données d'arêtes sont attribuées à différents fichiers en fonction d'une combinaison de l'étiquette de l'arête et des étiquettes des nœuds de début et de fin de l'arête.
- **`statements/`** : ce dossier contient des fichiers de données RDF au format Turtle, N-Quads, N-Triples ou JSON.
- **`config.json`** : ce fichier contient le schéma du graphe tel qu'il a été déduit par le processus d'exportation.
- **`lastEventId.json`** : ce fichier contient les attributs `commitNum` et `opNum` du dernier événement au niveau des flux Neptune de la base de données. Le processus d'exportation inclut uniquement ce fichier si vous définissez le paramètre d'exportation `includeLastEventId` sur `true`, et si les flux Neptune sont activés dans la base de données à partir de laquelle vous exportez les données.

Paramètres utilisés pour contrôler le processus d'exportation de Neptune

Que vous ayez recours au service d'exportation Neptune ou à l'utilitaire de ligne de commande `neptune-export`, les paramètres que vous utilisez pour contrôler l'exportation sont généralement

les mêmes. Ils contiennent un objet JSON transmis au point de terminaison d'exportation Neptune ou à `neptune-export` au niveau de la ligne de commande.

L'objet transmis au processus d'exportation comporte les champs de premier niveau suivants :

```
-d '{
      "command" : "(either export-pg or export-rdf)",
      "outputS3Path" : "s3:/(your Amazon S3 bucket)/(path to the folder for exported
 data)",
      "expectedBucketOwner" : "(your AWS account ID for Amazon S3 bucket)",
      "jobSize" : "(for Neptune-Export service only)",
      "params" : { (a JSON object that contains export-process parameters) },
      "additionalParams": { (a JSON object that contains parameters for training
 configuration) }
   }'
```

Table des matières

- Paramètre command
- Paramètre outputS3Path
- Paramètre expectedBucketOwner
- Paramètre jobSize
- Objet params.
- Objet additionalParams.
- Exportation des champs de paramètres dans l'objet JSON params de niveau supérieur
  - Liste des champs possibles dans l'objet params des paramètres d'exportation
    - Liste des champs communs à tous les types d'exportation
    - Liste des champs pour les exportations de graphes de propriétés
    - Liste des champs pour les exportations RDF
  - Champs communs à tous les types d'exportation
    - Champ cloneCluster dans params
    - Champ cloneClusterInstanceType dans params
    - Champ cloneClusterReplicaCount dans params
    - Champ cloneClusterEnableAuditLogs dans params
    - Champ clusterId dans params
    - Champ endpoint dans params

        - Champ endpoints dans params
        - Champ profile dans params
        - Champ useIamAuth dans params
        - Champ includeLastEventId dans params
    - Champs pour l'exportation du graphe de propriétés
        - Champ concurrency dans params
        - Champ edgeLabels dans params
        - Champ filter dans params
        - Champ filterConfigFile dans params
        - Champ format utilisé pour les données du graphe de propriétés dans params
        - Champ gremlinFilter dans params
        - Champ gremlinNodeFilter dans params
        - Champ gremlinEdgeFilter dans params
        - Champ nodeLabels dans params
        - Champ scope dans params
    - Champs pour l'exportation RDF
        - Champ format utilisé pour les données RDF dans params
        - Champ rdfExportScope dans params
        - Champ sparql dans params
        - Champ namedGraph dans params
- Exemples de filtrage des données exportées
    - Filtrage de l'exportation des données du graphe de propriétés
        - Exemple d'utilisation de scope pour exporter uniquement les arêtes
        - Exemple d'utilisation de nodeLabels et edgeLabels pour exporter uniquement les nœuds et les arêtes dotés d'étiquettes spécifiques
        - Exemple d'utilisation de filter pour exporter uniquement des nœuds, des arêtes et des propriétés spécifiés
        - Cet exemple utilise gremlinFilter.
        - Cet exemple utilise gremlinNodeFilter.
        - Cet exemple utilise gremlinEdgeFilter.
        - Combinaison de filter, gremlinNodeFilter, nodeLabels, edgeLabels et scope

- Filtrage de l'exportation des données RDF
  - Utilisation de rdfExportScope et sparql pour exporter des arêtes spécifiques
  - Utilisation namedGraph pour exporter un seul graphe nommé

Paramètre **command**

Le paramètre de niveau supérieur `command` détermine s'il faut exporter les données du graphe de propriétés ou les données RDF. Si vous omettez le paramètre `command`, le processus d'exportation exporte par défaut les données du graphe de propriétés.

- **`export-pg`** : exporte les données du graphe de propriétés.
- **`export-rdf`** : exporte les données RDF.

Paramètre **outputS3Path**

Le paramètre de niveau supérieur `outputS3Path` est obligatoire et doit contenir l'URI d'un emplacement Amazon S3 sur lequel les fichiers exportés peuvent être publiés :

```
  "outputS3Path" : "s3://(your Amazon S3 bucket)/(path to output folder)"
```

Cette valeur doit commencer par `s3://`, suivi d'un nom de compartiment valide et éventuellement d'un chemin de dossier au sein du compartiment.

Paramètre **expectedBucketOwner**

Le paramètre `expectedBucketOwner` de niveau supérieur est facultatif. Lorsqu'il est défini AWS sur un ID de compte, il confirme que le compartiment Amazon S3 spécifié dans `outputS3Path` appartient à ce compte. S'il est omis, il utilise par défaut l'ID de compte résolu à partir des informations d'identification utilisées pour les opérations Amazon S3.

```
  "expectedBucketOwner" : "(your AWS account ID for Amazon S3 bucket)"
```

Paramètre **jobSize**

Le paramètre de niveau supérieur `jobSize` est uniquement utilisé avec le service d'exportation Neptune, et non avec l'utilitaire de ligne de commande `neptune-export`. Il est facultatif. Il

vous permet de caractériser la taille de la tâche d'exportation que vous lancez, ce qui permet de déterminer la quantité de ressources de calcul qui seront consacrées à la tâche et son niveau de simultanéité maximal.

```
  "jobSize" : "(one of four size descriptors)"
```

Voici les quatre descripteurs de taille valides :

- `small` : simultanéité maximale de 8. Convient aux volumes de stockage pouvant atteindre 10 Go.
- `medium` : simultanéité maximale de 32. Convient aux volumes de stockage pouvant atteindre 100 Go.
- `large` : simultanéité maximale de 64. Convient aux volumes de stockage supérieurs à 100 Go, mais inférieurs à 1 To.
- `xlarge` : simultanéité maximale de 96. Convient aux volumes de stockage supérieurs à 1 To.

Par défaut, une exportation initiée sur le service d'exportation Neptune s'exécute comme une tâche `small`.

Les performances d'une exportation dépendent non seulement du paramètre `jobSize`, mais également du nombre d'instances de base de données à partir desquelles vous exportez des données, de la taille de chaque instance et du niveau de simultanéité effectif de la tâche.

Pour les exportations de graphes de propriétés, vous pouvez configurer le nombre d'instances de base de données à l'aide du paramètre cloneClusterReplicaCompter, et vous pouvez configurer le niveau de simultanéité effectif de la tâche à l'aide du paramètre simultanéité.

Objet **`params`**.

Le paramètre de niveau supérieur `params` est un objet JSON qui contient les paramètres que vous pouvez utiliser pour contrôler le processus d'exportation lui-même, comme expliqué dans Exportation des champs de paramètres dans l'objet JSON `params` de niveau supérieur. Certains champs de l'objet `params` sont spécifiques aux exportations de graphes de propriétés, d'autres auz données RDF.

Objet **`additionalParams`**.

Le paramètre de niveau supérieur `additionalParams` est un objet JSON qui contient les paramètres que vous pouvez utiliser pour contrôler les actions appliquées aux données après leur

exportation. Actuellement, `additionalParams` est uniquement utilisé pour exporter les données d'entraînement pour Neptune ML.

Exportation des champs de paramètres dans l'objet JSON **`params`** de niveau supérieur

L'objet JSON `params` d'exportation Neptune vous permet de contrôler l'exportation, y compris le type et le format des données exportées.

Liste des champs possibles dans l'objet **`params`** des paramètres d'exportation

Vous trouverez ci-dessous la liste de tous les champs de niveau supérieur qui peuvent apparaître dans un objet `params`. Seul un sous-ensemble de ces champs apparaît dans chaque objet.

Liste des champs communs à tous les types d'exportation

- `cloneCluster`
- `cloneClusterInstanceType`
- `cloneClusterReplicaCount`
- `cloneClusterEnableAuditLogs`
- `clusterId`
- `endpoint`
- `endpoints`
- `profile`
- `useIamAuth`
- `includeLastEventId`

Liste des champs pour les exportations de graphes de propriétés

- `concurrency`
- `edgeLabels`
- `filter`
- `filterConfigFile`
- `gremlinFilter`
- `gremlinNodeFilter`
- `gremlinEdgeFilter`
- `format`

- nodeLabels
- scope

Liste des champs pour les exportations RDF

- format
- rdfExportScope
- sparql
- namedGraph

Champs communs à tous les types d'exportation

Champ **`cloneCluster`** dans **`params`**

(Facultatif) Valeur par défaut : `false`.

Si le paramètre `cloneCluster` est défini sur `true`, le processus d'exportation utilise un clone rapide du cluster de bases de données :

```
  "cloneCluster" : true
```

Par défaut, le processus d'exportation exporte les données du cluster de bases de données que vous spécifiez à l'aide des paramètres `endpoint`, `endpoints` ou `clusterId`. Toutefois, si ce cluster de bases de données est utilisé pendant l'exportation et que les données changent, le processus d'exportation ne peut pas garantir la cohérence des données exportées.

Pour garantir la cohérence des données exportées, utilisez plutôt le paramètre `cloneCluster` afin d'exporter les données à partir d'un clone statique du cluster de bases de données.

Le cluster de bases de données cloné est créé dans le même VPC que le cluster de bases de données source. Il hérite des paramètres d'authentification du groupe de sécurité, du groupe de sous-réseaux et de la base de données IAM de la source. Lorsque l'exportation est terminée, Neptune supprime le cluster de bases de données cloné.

Par défaut, un cluster de bases de données cloné est constitué d'une instance unique de même type que l'instance principale du cluster de bases de données source. Vous pouvez modifier le type d'instance utilisé pour le cluster de bases de données cloné en en spécifiant un autre à l'aide de `cloneClusterInstanceType`.

> **Note**
>
> Si vous n'avez pas recours à l'option `cloneCluster` et que vous exportez directement les données à partir du cluster de bases de données principal, vous devrez peut-être augmenter le délai d'expiration des instances à partir desquelles les données sont exportées. Pour les jeux de données volumineux, le délai d'expiration devrait être fixé à plusieurs heures.

Champ **`cloneClusterInstanceType`** dans **`params`**

(Facultatif)

Si le paramètre `cloneCluster` est présent et défini sur `true`, vous pouvez recourir au paramètre `cloneClusterInstanceType` pour spécifier le type d'instance utilisé pour le cluster de bases de données cloné :

Par défaut, un cluster de bases de données cloné est constitué d'une instance unique de même type que l'instance principale du cluster de bases de données source.

```
  "cloneClusterInstanceType" : "(for example, r5.12xlarge)"
```

Champ **`cloneClusterReplicaCount`** dans **`params`**

(Facultatif)

Si le paramètre `cloneCluster` est présent et défini sur `true`, vous pouvez utiliser le paramètre `cloneClusterReplicaCount` pour spécifier le nombre d'instances de réplica en lecture créées dans le cluster de bases de données cloné :

```
  "cloneClusterReplicaCount" : (for example, 3)
```

Par défaut, un cluster de bases de données cloné se compose d'une seule instance principale. Le paramètre `cloneClusterReplicaCount` vous permet de spécifier le nombre d'instances de réplica en lecture supplémentaires à créer.

Champ **`cloneClusterEnableAuditLogs`** dans **`params`**

(Facultatif) Par défaut : false.

Si le `cloneCluster` paramètre est présent et défini sur true, vous pouvez l'`cloneClusterEnableAuditLogs`utiliser pour activer ou désactiver les journaux d'audit dans le cluster cloné.

Par défaut, la journalisation des audits est désactivée.

```
"cloneClusterEnableAuditLogs" : true
```

Champ **`clusterId`** dans **`params`**

(Facultatif)

Le paramètre `clusterId` spécifie l'ID d'un cluster de bases de données à utiliser :

```
  "clusterId" : "(the ID of your DB cluster)"
```

Si vous utilisez le paramètre `clusterId`, le processus d'exportation utilise toutes les instances disponibles dans ce cluster de bases de données pour extraire les données.

> **Note**
>
> Les paramètres `endpoint`, `endpoints` et `clusterId` s'excluent mutuellement. N'en utilisez qu'un seul à la fois.

Champ **`endpoint`** dans **`params`**

(Facultatif)

Utilisez `endpoint` pour spécifier un point de terminaison d'une instance Neptune dans le cluster de bases de données que le processus d'exportation peut interroger pour extraire des données (voir Connexions de point de terminaison). Il s'agit uniquement du nom DNS. Il n'inclut ni le protocole, ni le port :

```
  "endpoint" : "(a DNS endpoint of your DB cluster)"
```

Utilisez un point de terminaison de cluster ou d'instance, mais pas le point de terminaison de lecteur principal.

> **Note**
>
> Les paramètres `endpoint`, `endpoints` et `clusterId` s'excluent mutuellement. N'en utilisez qu'un seul à la fois.

Champ **`endpoints`** dans **`params`**

(Facultatif)

Utilisez `endpoints` pour spécifier un tableau JSON de points de terminaison dans le cluster de bases de données que le processus d'exportation peut interroger pour extraire des données (voir Connexions de point de terminaison). Il s'agit uniquement de noms DNS. Ils n'incluent ni le protocole, ni le port :

```
"endpoints": [
  "(one endpoint in your DB cluster)",
  "(another endpoint in your DB cluster)",
  "(a third endpoint in your DB cluster)"
  ]
```

Si le cluster comporte plusieurs instances (une instance principale et un ou plusieurs réplicas en lecture), vous pouvez améliorer les performances d'exportation en utilisant le paramètre `endpoints` pour répartir les requêtes sur une liste de ces points de terminaison.

> **Note**
>
> Les paramètres `endpoint`, `endpoints` et `clusterId` s'excluent mutuellement. N'en utilisez qu'un seul à la fois.

Champ **`profile`** dans **`params`**

(Nécessaire pour exporter les données d'entraînement pour Neptune ML, sauf si le champ *`neptune_ml`* se trouve dans le champ *`additionalParams`*).

Le paramètre `profile` fournit des ensembles de paramètres préconfigurés pour des charges de travail spécifiques. À l'heure actuelle, le processus d'exportation ne prend en charge que le profil `neptune_ml`.

Si vous exportez des données d'entraînement pour Neptune ML, ajoutez le paramètre suivant à l'objet `params` :

```
"profile" : "neptune_ml"
```

Champ **`useIamAuth`** dans **`params`**

(Facultatif) Valeur par défaut : `false`.

Si l'authentification IAM est activée dans la base de données à partir de laquelle vous exportez les données, vous devez inclure le paramètre `useIamAuth` défini sur `true` :

```
"useIamAuth" : true
```

Champ **`includeLastEventId`** dans **`params`**

Si vous définissez `includeLastEventId` sur true et que Neptune Streams est activé dans la base de données à partir de laquelle vous exportez les données, le processus d'exportation écrit un fichier `lastEventId.json` à l'emplacement d'exportation que vous avez indiqué. Ce fichier contient les valeurs `commitNum` et opNum du dernier événement du flux.

```
"includeLastEventId" : true
```

Une base de données clonée créée par le processus d'exportation hérite du paramètre de flux de son parent. Si les flux sont activés pour le parent, ils le sont aussi pour le clone. Le contenu du flux sur le clone reflétera le contenu du parent (y compris le même événement IDs) au moment où le clone a été créé.

Champs pour l'exportation du graphe de propriétés

Champ **`concurrency`** dans **`params`**

(Facultatif) Valeur par défaut : 4.

Le paramètre `concurrency` indique le nombre de requêtes parallèles que le processus d'exportation doit utiliser :

```
"concurrency" : (for example, 24)
```

Il est conseillé de définir le niveau de simultanéité à deux fois le nombre de v CPUs sur toutes les instances à partir desquelles vous exportez des données. Une instance r5.xlarge, par exemple,

possède 4 v. CPUs Si vous exportez à partir d'un cluster de 3 instances r5.xlarge, vous pouvez définir le niveau de simultanéité sur 24 (= 3 x 2 x 4).

Si vous utilisez le service d'exportation Neptune, le niveau de simultanéité est limité par le paramètre jobSize. Par exemple, une tâche de petite envergure prend en charge le niveau de simultanéité 8. Si vous essayez de spécifier le niveau de simultanéité 24 pour une tâche de petite envergure à l'aide du paramètre `concurrency`, le niveau de simultanéité 8 sera appliqué.

Si vous exportez des données à partir d'un cluster cloné, le processus d'exportation calcule un niveau de simultanéité approprié en fonction de la taille des instances clonées et de la taille de la tâche.

Champ **`edgeLabels`** dans **`params`**

(Facultatif)

Utilisez `edgeLabels` pour exporter uniquement les arêtes dont vous spécifiez les étiquettes :

```
  "edgeLabels" : ["(a label)", "(another label"]
```

Chaque étiquette du tableau JSON doit être une étiquette simple et unique.

Le paramètre `scope` a priorité sur le paramètre `edgeLabels`. Ainsi, si la valeur `scope` n'inclut pas d'arêtes, le paramètre `edgeLabels` n'a aucun effet.

Champ **`filter`** dans **`params`**

(Facultatif)

`filter`À utiliser pour spécifier que seules and/or les arêtes des nœuds portant des étiquettes spécifiques doivent être exportées et pour filtrer les propriétés exportées pour chaque nœud ou arête.

La structure générale d'un objet `filter`, qu'il soit intégré directement ou dans un fichier de configuration de filtre, est la suivante :

```
  "filter" : {
    "nodes": [ (array of node label and properties objects) ],
    "edges": [ (array of edge definition an properties objects) ]
  }
```

- **`nodes`** : contient un tableau JSON de nœuds et de propriétés de nœuds sous la forme suivante :

```
    "nodes : [
```

```
    {
      "label": "(node label)",
      "properties": [ "(a property name)", "(another property name)", ( ... ) ]
    }
  ]
```

- `label` : représente la ou les étiquettes du graphe de propriétés du nœud.

  Utilise une valeur unique ou, si le nœud possède plusieurs étiquettes, un tableau de valeurs.
- `properties` : contient un tableau des noms des propriétés de nœud que vous souhaitez exporter.

- **`edges`** : contient un tableau JSON de définitions d'arêtes sous la forme suivante :

```
  "edges" : [
    {
      "label": "(edge label)",
      "properties": [ "(a property name)", "(another property name)", ( ... ) ]
    }
  ]
```

- `label` : étiquette de graphe de propriétés de l'arête. Utilise une seule valeur.
- `properties` : contient un tableau des noms des propriétés de l'arête, que vous souhaitez exporter.

Champ **`filterConfigFile`** dans **`params`**

(Facultatif)

Utilisez `filterConfigFile` pour spécifier un fichier JSON contenant une configuration de filtre sous la même forme que le paramètre `filter` :

```
 "filterConfigFile" : "s3://(your Amazon S3 bucket)/neptune-export/(the name of the
JSON file)"
```

Consultez filtre pour le format du fichier `filterConfigFile`.

Champ **`format`** utilisé pour les données du graphe de propriétés dans **`params`**

(Facultatif) Par défaut : `csv` (valeurs séparées par des virgules)

Le paramètre `format` spécifie le format de sortie des données exportées du graphe de propriétés :

```
"format" : (one of: csv, csvNoHeaders, json, neptuneStreamsJson)
```

- **`csv`** : sortie au format CSV (valeurs séparées par des virgules), avec une mise en forme des en-têtes de colonnes conformément au format de données de chargement Gremlin.
- **`csvNoHeaders`** : données au format CSV sans en-têtes de colonne.
- **`json`** : données au format JSON.
- **`neptuneStreamsJson`** : données au format JSON qui utilisent le format de sérialisation des modifications GREMLIN_JSON.

Champ **`gremlinFilter`** dans **`params`**

(Facultatif)

Le paramètre `gremlinFilter` vous permet de fournir un extrait Gremlin, tel qu'une étape `has()`, qui permet de filtrer à la fois les nœuds et les arêtes :

```
"gremlinFilter" : (a Gremlin snippet)
```

Les noms de champs et les valeurs de chaîne doivent être entourés de guillemets doubles précédés d'un caractère d'échappement. Pour les dates et les heures, vous pouvez utiliser la méthode datetime.

L'exemple suivant exporte uniquement les nœuds et les arêtes dont la valeur de la propriété de date de création est ultérieure au 10/10/2021 :

```
"gremlinFilter" : "has(\"created\", gt(datetime(\"2021-10-10\")))"
```

Champ **`gremlinNodeFilter`** dans **`params`**

(Facultatif)

Le paramètre `gremlinNodeFilter` vous permet de fournir un extrait Gremlin, tel qu'une étape `has()`, qui permet de filtrer les nœuds :

```
"gremlinNodeFilter" : (a Gremlin snippet)
```

Les noms de champs et les valeurs de chaîne doivent être entourés de guillemets doubles précédés d'un caractère d'échappement. Pour les dates et les heures, vous pouvez utiliser la méthode datetime.

L'exemple suivant exporte uniquement les nœuds dont la valeur d'une propriété booléenne `deleted` est `true` :

```
  "gremlinNodeFilter" : "has(\"deleted\", true)"
```

Champ **`gremlinEdgeFilter`** dans **`params`**

(Facultatif)

Le paramètre `gremlinEdgeFilter` vous permet de fournir un extrait Gremlin, tel qu'une étape `has()`, qui permet de filtrer les arêtes :

```
  "gremlinEdgeFilter" : (a Gremlin snippet)
```

Les noms de champs et les valeurs de chaîne doivent être entourés de guillemets doubles précédés d'un caractère d'échappement. Pour les dates et les heures, vous pouvez utiliser la méthode datetime.

L'exemple suivant exporte uniquement les arêtes dont la valeur d'une propriété numérique `strength` est 5 :

```
  "gremlinEdgeFilter" : "has(\"strength\", 5)"
```

Champ **`nodeLabels`** dans **`params`**

(Facultatif)

Utilisez `nodeLabels` pour exporter uniquement les nœuds dont vous spécifiez les étiquettes :

```
  "nodeLabels" : ["(a label)", "(another label"]
```

Chaque étiquette du tableau JSON doit être une étiquette simple et unique.

Le paramètre `scope` a priorité sur le paramètre `nodeLabels`. Ainsi, si la valeur `scope` n'inclut pas de nœuds, le paramètre `nodeLabels` n'a aucun effet.

Champ **`scope`** dans **`params`**

(Facultatif) Valeur par défaut : `all`.

Le paramètre `scope` indique si seuls les nœuds, seules les arêtes ou à la fois les nœuds et les arêtes doivent être exportés :

```
  "scope" : (one of: nodes, edges, or all)
```

- `nodes` : exporte uniquement les nœuds et leurs propriétés.
- `edges` : exporte uniquement les arêtes et leurs propriétés.
- `all` : exporte à la fois les nœuds et les arêtes ainsi que leurs propriétés (par défaut).

Champs pour l'exportation RDF

Champ **`format`** utilisé pour les données RDF dans **`params`**

(Facultatif) Par défaut : `turtle`

Le paramètre `format` spécifie le format de sortie des données RDF exportées :

```
  "format" : (one of: turtle, nquads, ntriples, neptuneStreamsJson)
```

- **`turtle`** : sortie au format Turtle.
- **`nquads`** : données au format N-Quads sans en-têtes de colonne.
- **`ntriples`** : données au format N-Triples.
- **`neptuneStreamsJson`** : données au format JSON qui utilisent le format de sérialisation des modifications SPARQL NQUADS.

Champ **`rdfExportScope`** dans **`params`**

(Facultatif) Valeur par défaut : `graph`.

Le paramètre `rdfExportScope` indique la portée de l'exportation RDF :

```
  "rdfExportScope" : (one of: graph, edges, or query)
```

- `graph` : exporte toutes les données RDF.

- `edges` : exporte uniquement les triplets qui représentent des arêtes.
- `query` : exporte les données récupérées par une requête SPARQL fournie à l'aide du champ `sparql`.

Champ **`sparql`** dans **`params`**

(Facultatif)

Le paramètre `sparql` permet de spécifier une requête SPARQL pour récupérer les données à exporter :

```
  "sparql" : (a SPARQL query)
```

Si vous fournissez une requête à l'aide du champ `sparql`, vous devez également définir le champ `rdfExportScope` sur `query`.

Champ **`namedGraph`** dans **`params`**

(Facultatif)

Le `namedGraph` paramètre vous permet de spécifier un IRI pour limiter l'exportation à un seul graphe nommé :

```
  "namedGraph" : (Named graph IRI)
```

Le `namedGraph` paramètre ne peut être utilisé qu'avec le `rdfExportScope` champ défini sur`graph`.

Exemples de filtrage des données exportées

Voici des exemples illustrant les méthodes de filtrage des données exportées.

Filtrage de l'exportation des données du graphe de propriétés

Exemple d'utilisation de **`scope`** pour exporter uniquement les arêtes

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "scope": "edges"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Exemple d'utilisation de **`nodeLabels`** et **`edgeLabels`** pour exporter uniquement les nœuds et les arêtes dotés d'étiquettes spécifiques

Le paramètre `nodeLabels` de l'exemple suivant indique que seuls les nœuds dotés d'une étiquette `Person` ou `Post` doivent être exportés. Le paramètre `edgeLabels` indique que seules les arêtes dotées d'une étiquette `likes` doivent être exportées :

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "nodeLabels": ["Person", "Post"],
    "edgeLabels": ["likes"]
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Exemple d'utilisation de **`filter`** pour exporter uniquement des nœuds, des arêtes et des propriétés spécifiés

Dans cet exemple, l'objet `filter` exporte les nœuds `country` avec leurs propriétés `type`, `code` et `desc`, ainsi que les arêtes `route` avec leurs propriétés `dist`.

```
{
```

```
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "filter": {
      "nodes": [
        {
          "label": "country",
          "properties": [
            "type",
            "code",
            "desc"
          ]
        }
      ],
      "edges": [
        {
          "label": "route",
          "properties": [
            "dist"
          ]
        }
      ]
    }
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Cet exemple utilise **`gremlinFilter`**.

Cet exemple utilise `gremlinFilter` pour exporter uniquement les nœuds et les arêtes créés après le 10/10/2021 (c'est-à-dire avec une propriété `created` dont la valeur est ultérieure au 10/10/2021) :

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinFilter" : "has(\"created\", gt(datetime(\"2021-10-10\")))"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Cet exemple utilise **`gremlinNodeFilter`**.

Cet exemple utilise `gremlinNodeFilter` pour exporter uniquement les nœuds supprimés (nœuds dotés d'une propriété booléenne `deleted` dont la valeur est `true`) :

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinNodeFilter" : "has(\"deleted\", true)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Cet exemple utilise **`gremlinEdgeFilter`**.

Cet exemple utilise `gremlinEdgeFilter` pour exporter uniquement les arêtes ayant une propriété numérique `strength` dont la valeur est 5 :

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinEdgeFilter" : "has(\"strength\", 5)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Combinaison de **`filter`**, **`gremlinNodeFilter`**, **`nodeLabels`**, **`edgeLabels`** et **`scope`**

Dans cet exemple, l'objet `filter` exporte les éléments suivants :

- Nœuds `country` avec leurs propriétés `type`, `code` et `desc`
- Nœuds `airport` avec leurs propriétés `code`, `icao` et `runways`
- Arêtes `route` avec leur propriété `dist`

Le paramètre `gremlinNodeFilter` filtre les nœuds afin que seuls ceux dont la propriété `code` commence par A soient exportés.

Les paramètres `nodeLabels` et `edgeLabels` limitent davantage la sortie, de sorte que seuls les nœuds `airport` et les arêtes `route` sont exportés.

Enfin, le paramètre `scope` élimine les arêtes lors de l'exportation, ce qui ne laisse que les nœuds `airport` désignés dans la sortie.

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "filter": {
      "nodes": [
        {
          "label": "airport",
          "properties": [
            "code",
            "icao",
            "runways"
          ]
        },
        {
          "label": "country",
          "properties": [
            "type",
            "code",
            "desc"
          ]
        }
      ],
      "edges": [
        {
          "label": "route",
          "properties": [
            "dist"
          ]
        }
      ]
    },
    "gremlinNodeFilter": "has(\"code\", startingWith(\"A\"))",
    "nodeLabels": [
      "airport"
    ],
    "edgeLabels": [
      "route"
    ],
    "scope": "nodes"
```

```
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Filtrage de l'exportation des données RDF

Utilisation de **`rdfExportScope`** et **`sparql`** pour exporter des arêtes spécifiques

Cet exemple exporte des triplets dont le prédicat est < http://kelvinlawrence. net/air-routes/objectProperty/route> et dont l'objet n'est pas un littéral :

```
{
  "command": "export-rdf",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "rdfExportScope": "query",
    "sparql": "CONSTRUCT { ?s <http://kelvinlawrence.net/air-routes/objectProperty/
route> ?o } WHERE { ?s ?p ?o . FILTER(!isLiteral(?o)) }"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Utilisation **`namedGraph`** pour exporter un seul graphe nommé

Cet exemple exporte des triplets appartenant au graphe nommé < http://aws.amazon. com/neptune/vocab/v01/DefaultNamedGraph> :

```
{
  "command": "export-rdf",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "rdfExportScope": "graph",
    "namedGraph": "http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Résolution des problèmes liés au processus d'exportation de Neptune

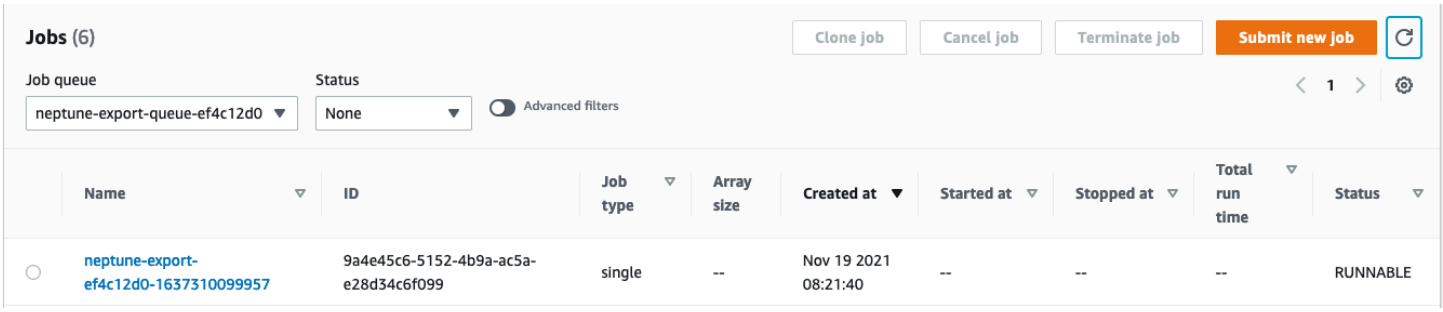
Le processus d'exportation d'Amazon Neptune utilise AWS Batch pour provisionner les ressources de calcul et de stockage nécessaires à l'exportation des données Neptune. Lorsqu'une exportation est en cours, vous pouvez utiliser le lien dans le `logs` champ pour accéder aux CloudWatch journaux de la tâche d'exportation.

Toutefois, les CloudWatch journaux de la AWS Batch tâche qui effectue l'exportation ne sont disponibles que lorsque la AWS Batch tâche est en cours d'exécution. Si l'exportation Neptune indique qu'une exportation est en attente, il n'y aura pas de lien permettant d'accéder aux CloudWatch journaux. Si une tâche d'exportation reste en l'`pending`état pendant plus de quelques minutes, un problème peut survenir lors du provisionnement des AWS Batch ressources sous-jacentes.

Lorsque la tâche d'exportation n'est plus en attente, vous pouvez vérifier son statut comme suit :

Pour vérifier le statut d'une AWS Batch tâche

1. Ouvrez la AWS Batch console à l'adresse https://console.aws.amazon.com/batch/.
2. Sélectionnez la file d'attente des tâches d'exportation de Neptune.
3. Recherchez la tâche dont le nom correspond au `jobName` renvoyé par Neptune lorsque vous avez commencé l'exportation.

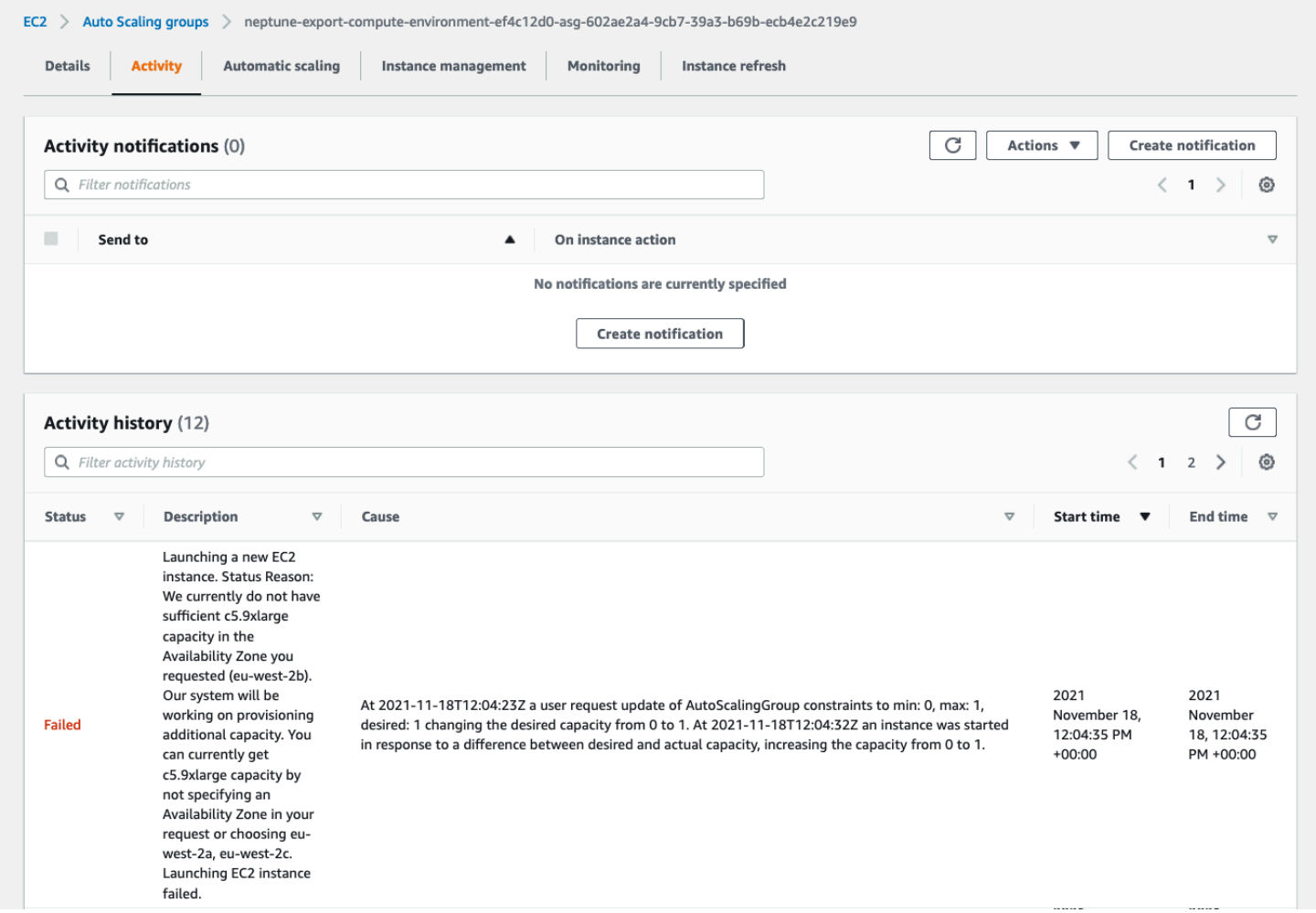


Si la tâche reste bloquée à l'état `RUNNABLE`, cela est peut-être dû au fait que des problèmes de réseau ou de sécurité empêchent l'instance de conteneur de communiquer avec le cluster Amazon Elastic Container Service (Amazon ECS) sous-jacent. Consultez la section consacrée à la vérification des paramètres réseau et de sécurité de l'environnement informatique dans cet article de support.

Vous pouvez également vérifier les problèmes liés à l'autoscaling :

Pour vérifier le groupe Amazon EC2 auto-scaling pour l'environnement de calcul AWS Batch

1. Ouvrez la EC2 console Amazon à l'adresse https://console.aws.amazon.com/ec2/.
2. Sélectionnez le groupe Auto Scaling correspondant à l'environnement de calcul neptune-export.
3. Ouvrez l'onglet Activité et vérifiez l'historique des activités pour détecter les événements ayant échoué.



Erreurs courantes d'exportation Neptune

`org.eclipse.rdf4j.query.QueryEvaluationException: Tag mismatch!`

Si une tâche `export-rdf` échoue régulièrement avec une exception `QueryEvaluationException Tag mismatch!`, l'instance Neptune est sous-dimensionnée pour les requêtes volumineuses et de longue durée utilisées par Neptune-Export.

Vous pouvez éviter cette erreur en passant à une instance Neptune plus grande ou en configurant la tâche pour qu'elle soit exportée à partir d'un cluster cloné de grande taille, comme suit :

```
'{
  "command": "export-rdf",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "cloneCluster": True,
    "cloneClusterInstanceType" : "r5.24xlarge"
  }
}'
```

Exportation des résultats de requêtes Gremlin vers Amazon S3

À partir de la version 1.4.3.0 du moteur, Amazon Neptune prend en charge l'exportation des résultats des requêtes Gremlin directement vers Amazon S3. Cette fonctionnalité vous permet de gérer efficacement les résultats de requêtes volumineux en les exportant vers un compartiment Amazon S3 au lieu de les renvoyer sous forme de réponse à une requête.

Pour exporter les résultats d'une requête vers Amazon S3, utilisez l'`call()`étape portant le nom du `neptune.query.exportToS3` service comme dernière étape de votre requête Gremlin. L'étape du terminal dans les pilotes Tinkerpop utilisant le Bytecode peut être ajoutée après l'étape. `call()` Les paramètres d'exportation doivent être fournis sous forme de chaînes de caractères.

> **Note**
>
> La requête comportant l'`call()`étape `neptune.query.exportToS3` échouera si elle n'est pas utilisée comme étape finale. Les clients Gremlin utilisant le bytecode peuvent utiliser les étapes du terminal. Pour plus d'informations, consultez les meilleures pratiques relatives à Gremlin dans la documentation Amazon Neptune.

```
g.V()
  ...
  .call('neptune.query.exportToS3', [
    'destination': 's3://your-bucket/path/result.json',
    'format': 'GraphSONv3',
    'kmskeyArn': 'optional-kms-key-arn'
```

```
  ])
```

Parameters

- `destination`: obligatoire - L'URI Amazon S3 dans lequel les résultats seront écrits.
- `format`: obligatoire - Le format de sortie ne prend actuellement en charge que « Graph SONv3 ».
- `keyArn`: facultatif - L'ARN d'une AWS KMS clé pour le chiffrement côté serveur Amazon S3.

Exemples

Exemple de requête

```
g.V().
    hasLabel('Comment').
    valueMap().
    call('neptune.query.exportToS3', [
    'destination': 's3://your-bucket/path/result.json',
    'format': 'GraphSONv3',
    'keyArn': 'optional-kms-key-arn'
  ])
```

Exemple de réponse à une requête

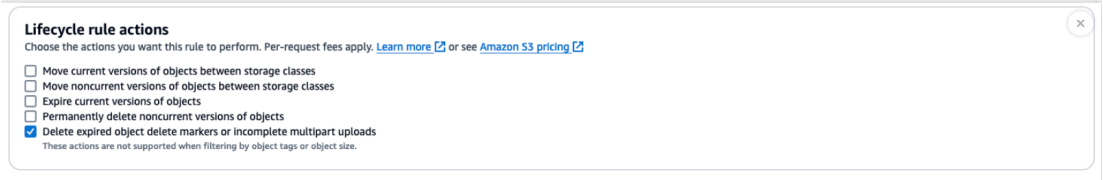
```
{
    "destination":"s3://your-bucket/path/result.json,
    "exportedResults": 100,
    "exportedBytes": 102400
}
```

Conditions préalables

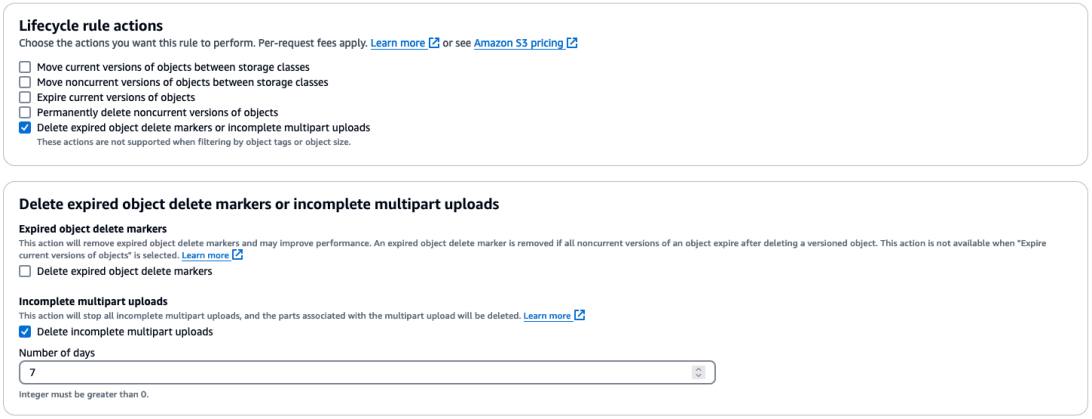
- Votre instance de base de données Neptune doit avoir accès à Amazon S3 via un point de terminaison VPC de type passerelle.
- Pour utiliser un AWS KMS chiffrement personnalisé dans la requête, un point de terminaison VPC de type interface est requis AWS KMS pour permettre à Neptune de communiquer avec. AWS KMS
- Vous devez activer l'authentification IAM sur Neptune et disposer des autorisations IAM appropriées pour écrire dans le compartiment Amazon S3 cible. Le fait de ne pas le faire

provoquera une erreur de 400 demandes erronées « Le cluster doit avoir l'authentification IAM activée pour S3 Export ».

- Le compartiment Amazon S3 cible :
  - Le compartiment Amazon S3 cible ne doit pas être public. `Block public access`doit être activé.
  - La destination Amazon S3 cible doit être vide.
  - Le compartiment Amazon S3 cible doit être doté d'une règle de cycle de vie `Delete expired object delete markers or incomplete multipart uploads` activée`Delete incomplete multipart uploads`. Consultez la mise à jour sur la gestion du cycle de vie d'Amazon S3 : prise en charge des téléchargements partitionnés et des marqueurs de suppression pour plus d'informations.



  - Le compartiment Amazon S3 cible doit être doté d'une règle de cycle de vie `Delete incomplete multipart uploads` définie sur `Delete expired object delete markers or incomplete multipart uploads` une valeur supérieure à celle que prendra l'évaluation de la requête (par exemple, 7 jours). Cela est nécessaire pour supprimer les téléchargements incomplets (qui ne sont pas directement visibles mais qui entraîneraient des coûts) au cas où Neptune ne pourrait pas les terminer ou les annuler (par exemple, en raison d'échecs). instance/engine Consultez la mise à jour sur la gestion du cycle de vie d'Amazon S3 : prise en charge des téléchargements partitionnés et des marqueurs de suppression pour plus d'informations.



Considérations importantes

- L'étape d'exportation doit être la dernière étape de votre requête Gremlin.
- Si un objet existe déjà à l'emplacement Amazon S3 spécifié, la requête échouera.
- La durée maximale d'exécution des requêtes d'exportation est limitée à 11 heures et 50 minutes. Cette fonctionnalité utilise des sessions d'accès direct. Il est actuellement limité à 11 heures et 50 minutes afin d'éviter les problèmes d'expiration des jetons.

> **Note**
>
> La requête d'exportation respecte toujours le délai d'expiration de la requête. Pour les exportations volumineuses, vous devez utiliser un délai d'attente de requête approprié.

- Tous les nouveaux téléchargements d'objets vers Amazon S3 sont automatiquement chiffrés.
- Pour éviter les coûts de stockage liés à des chargements partitionnés incomplets en cas d'erreur ou de panne, nous vous recommandons de configurer une règle de cycle de vie pour votre `Delete incomplete multipart uploads` compartiment Amazon S3.

Format de la réponse

Plutôt que de renvoyer directement les résultats de la requête, celle-ci renvoie des métadonnées relatives à l'opération d'exportation, notamment le statut et les détails de l'exportation. Les résultats de la requête dans Amazon S3 seront au SONv3 format graphique.

```
{
  "data": {
    "@type": "g:List",
    "@value": [
      {
        "@type": "g:Map",
        "@value": [
          "browserUsed",
          {
            "@type": "g:List",
            "@value": [
              "Safari"
            ]
          },
          "length",
```

```
          {
            "@type": "g:List",
            "@value": [
              {
                "@type": "g:Int32",
                "@value": 7
              }
            ]
          },
          "locationIP",
          {
            "@type": "g:List",
            "@value": [
              "192.0.2.0/24"
            ]
          },
          "creationDate",
          {
            "@type": "g:List",
            "@value": [
              {
                "@type": "g:Date",
                "@value": 1348341961000
              }
            ]
          },
          "content",
          {
            "@type": "g:List",
            "@value": [
              "no way!"
            ]
          }
        ]
      },
      {
        "@type": "g:Map",
        "@value": [
          "browserUsed",
          {
            "@type": "g:List",
            "@value": [
              "Firefox"
            ]
```

```
          },
          "length",
          {
            "@type": "g:List",
            "@value": [
              {
                "@type": "g:Int32",
                "@value": 2
              }
            ]
          },
          "locationIP",
          {
            "@type": "g:List",
            "@value": [
              "203.0.113.0/24"
            ]
          },
          "creationDate",
          {
            "@type": "g:List",
            "@value": [
              {
                "@type": "g:Date",
                "@value": 1348352960000
              }
            ]
          },
          "content",
          {
            "@type": "g:List",
            "@value": [
              "ok"
            ]
          }
        ]
      },


      ...


    ]
  }
```

```
}
```

Sécurité

- Toutes les données transférées vers Amazon S3 sont cryptées en transit à l'aide du protocole SSL.
- Vous pouvez spécifier une AWS KMS clé pour le chiffrement côté serveur des données exportées. Amazon S3 chiffre les nouvelles données par défaut. Si le bucket est configuré pour utiliser une AWS KMS clé spécifique, c'est cette clé qui est utilisée.
- Neptune vérifie que le compartiment cible n'est pas public avant de commencer l'exportation.
- Les exportations entre comptes et entre régions ne sont pas prises en charge.

Gestion des erreurs

- Le compartiment Amazon S3 cible est public.
- L'objet spécifié existe déjà.
- Vous ne disposez pas des autorisations suffisantes pour écrire dans le compartiment Amazon S3.
- L'exécution de la requête dépasse la limite de temps maximale.

Bonnes pratiques

- Utilisez les règles du cycle de vie des compartiments Amazon S3 pour nettoyer les téléchargements partitionnés incomplets.
- Surveillez vos opérations d'exportation à l'aide des journaux et des indicateurs de Neptune. Vous pouvez vérifier le point de terminaison d'état de Gremlin pour voir si une requête est en cours d'exécution. Tant que le client n'a pas reçu de réponse, la requête sera considérée comme étant en cours d'exécution.

Octroi de l'accès à la fonctionnalité d'exportation Amazon S3 de Gremlin

Politiques IAM requises

1. Accès en lecture aux requêtes Neptune

```
{
  "Sid": "NeptuneQueryRead",
  "Effect": "Allow",
```

```
  "Action": ["neptune-db:Read*"],
  "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-ABCD12/*"
}
```

Pourquoi c'est nécessaire : cette autorisation permet de lire les données des bases de données Neptune, ce qui est nécessaire pour exécuter les requêtes Gremlin qui seront exportées. L'exemple précédent autorise les requêtes de lecture. Pour une read/write requête, write/delete des autorisations sont requises.

2. Autorisations d'exportation Amazon S3

```
{
  "Sid": "NeptuneS3Export",
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:PutObject",
    "s3:AbortMultipartUpload",
    "s3:GetBucketPublicAccessBlock"
  ],
  "Resource": "arn:aws:s3:::neptune-export-bucket/*"
}
```

Pourquoi chaque autorisation est nécessaire :

- `s3:ListBucket`: obligatoire pour vérifier l'existence du bucket et le contenu de la liste.
- `s3:PutObject`: Nécessaire pour écrire les données exportées sur Amazon S3.
- `s3:AbortMultipartUpload`: Nécessaire pour nettoyer les téléchargements partitionnés incomplets en cas d'échec de l'exportation.
- `s3:GetBucketPublicAccessBlock`: Nécessaire comme mesure de sécurité pour vérifier que le bucket n'est pas public avant d'exporter des données.

3. AWS KMSautorisations - facultatives. Nécessaire uniquement si vous utilisez un AWS KMS chiffrement personnalisé.

```
{
  "Sid": "NeptuneS3ExportKMS",
  "Effect": "Allow",
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey",
```

```
    "kms:DescribeKey"
  ],
  "Resource": "arn:aws:kms:<REGION>:<AWS_ACCOUNT_ID>:key/mrk-48971c37"
    "Condition": {
    "StringEquals": {
      "kms:ViaService": [
        "s3.<REGION>.amazonaws.com",
        "rds.<REGION>.amazonaws.com"
      ]
    }
  }
}
```

Pourquoi chaque autorisation est nécessaire :

- `kms:Decrypt`: Nécessaire pour déchiffrer la AWS KMS clé pour le chiffrement des données.
- `kms:GenerateDataKey`: Nécessaire pour générer des clés de données afin de chiffrer les données exportées.
- `kms:DescribeKey`: Nécessaire pour vérifier et récupérer les informations relatives à la AWS KMS clé.
- `kms:ViaService`: augmente la sécurité en faisant en sorte que la clé ne soit pas utilisable par ce rôle pour aucun autre AWS service.

Conditions préalables importantes

- Authentification IAM : doit être activée sur le cluster Neptune pour appliquer ces autorisations.
- Point de terminaison VPC :
  - Un point de terminaison VPC de type Gateway pour Amazon S3 est nécessaire pour permettre à Neptune de communiquer avec Amazon S3.
  - Pour utiliser un AWS KMS chiffrement personnalisé dans la requête, un point de terminaison VPC de type interface est requis AWS KMS pour permettre à Neptune de communiquer avec. AWS KMS
- Configuration du compartiment Amazon S3 :
  - Ne doit pas être public.
  - Devrait avoir une règle de cycle de vie pour nettoyer les téléchargements partitionnés incomplets.
  - Chiffrera automatiquement les nouveaux objets.

Ces autorisations et conditions préalables garantissent une exportation sûre et fiable des résultats des requêtes Gremlin tout en maintenant des contrôles d'accès et des mesures de protection des données appropriés.

Gestion de votre base de données Amazon Neptune

Cette section explique comment gérer et maintenir votre cluster de base de données Neptune à l'aide du AWS Management Console et du. AWS CLI

Neptune opère sur des clusters de serveurs de base de données qui sont connectés dans une topologie de réplication. Par conséquent, la gestion de Neptune implique souvent de déployer des modifications sur plusieurs serveurs et de s'assurer que tous les réplicas Neptune suivent le rythme du serveur principal.

Étant donné que Neptune met à l'échelle le stockage sous-jacent en toute transparence à mesure que vos données augmentent, la gestion de Neptune exige relativement peu de gestion au niveau du stockage sur disque. De la même manière, étant donné que Neptune effectue automatiquement des sauvegardes continues, un cluster Neptune nécessite peu de planification ou de temps d'arrêt pour la réalisation des sauvegardes.

Rubriques

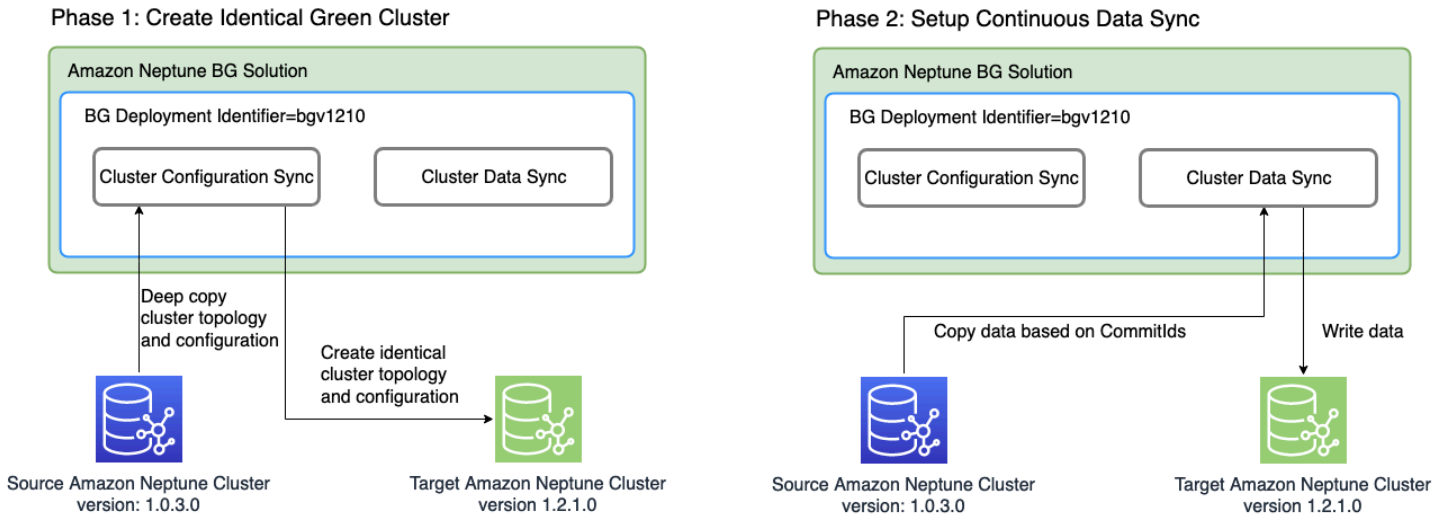
- Utilisation de la Blue/Green solution Neptune pour effectuer des mises à jour bleues
- Création d'un utilisateur IAM avec les autorisations nécessaires pour accéder à Neptune
- Groupes de paramètres Amazon Neptune
- Paramètres Amazon Neptune
- Lancement d'un cluster de base de données Neptune à l'aide du AWS Management Console
- Arrêt et démarrage d'un cluster de bases de données Amazon Neptune.
- Vider un cluster de bases de données Amazon Neptune à l'aide de l'API de réinitialisation rapide
- Ajout d'instances de lecteur Neptune à un cluster de bases de données
- Création d'une instance de lecteur Neptune à l'aide de la console
- Modification d'un cluster de bases de données Neptune à l'aide de la console
- Performances et mise à l'échelle dans Amazon Neptune
- Autoscaling du nombre de réplicas dans un cluster de bases de données Amazon Neptune
- Maintenance du cluster de bases de données Amazon Neptune
- Utilisation d'un CloudFormation modèle pour mettre à jour la version du moteur de votre cluster de base de données Neptune
- Clonage de bases de données dans Neptune
- Gestion des instances Amazon Neptune

Utilisation de la Blue/Green solution Neptune pour effectuer des mises à jour bleues

Les mises à niveau du moteur Amazon Neptune peuvent impliquer un temps d'arrêt des applications, car la base de données n'est pas disponible pendant l'installation et la vérification des mises à jour, et ce que les mises à niveau aient initiées manuellement ou automatiquement.

Neptune fournit une solution de Blue/Green déploiement que vous pouvez exécuter à l'aide d'une CloudFormation pile et qui réduit considérablement ces temps d'arrêt. Il crée un environnement intermédiaire vert qui est synchronisé avec l'environnement de production bleu. Vous pouvez ensuite mettre à jour cet environnement intermédiaire pour effectuer une mise à niveau mineure ou majeure de la version du moteur, une modification du modèle de données de graphe ou une mise à jour du système d'exploitation, puis tester le résultat. Enfin, vous pouvez le changer rapidement pour en faire votre environnement de production, avec un temps d'arrêt minime.

La Blue/Green solution Neptune passe par deux phases, comme illustré dans ce schéma :



La phase 1 crée un cluster de bases de données vert identique au cluster de production

La solution crée un cluster de base de données avec un identifiant de blue/green déploiement unique et avec la même topologie de cluster que votre cluster de production. En d'autres termes, il possède le même nombre et la même taille d'instances de base de données, les mêmes groupes de paramètres et les mêmes configurations que le cluster de bases de données de production (bleu), sauf qu'il a été mis à niveau vers la version de moteur cible que vous avez spécifiée, qui doit être supérieure à la version actuelle du moteur (bleu). Vous pouvez spécifier une version mineure et une version majeure du moteur pour la cible. Si nécessaire, la solution effectue toutes les mises à niveau

intermédiaires nécessaires pour atteindre la version cible spécifiée du moteur. Ce nouveau cluster devient l'environnement intermédiaire vert.

La phase 2 configure la synchronisation continue des données

Une fois que l'environnement vert a été entièrement préparé, la solution configure une réplication continue entre le cluster source (bleu) et le cluster cible (vert) à l'aide des flux Neptune. Lorsque la différence de réplication entre eux atteint zéro, l'environnement intermédiaire est prêt à être testé. À ce stade, vous devez suspendre l'écriture dans le cluster bleu pour éviter tout retard de réplication supplémentaire.

La version cible du moteur peut avoir de nouvelles fonctionnalités ou dépendances qui affectent vos applications. Consultez la page de la version cible du moteur et les pages de mises à jour apportées entre-temps au moteur sous Versions du moteur pour identifier ce qui a changé depuis la version actuelle du moteur. Il est préférable d'exécuter des tests d'intégration ou de vérifier vos applications manuellement sur le cluster vert avant de le promouvoir dans l'environnement de production.

Après avoir testé et qualifié les modifications dans le cluster vert, il vous suffit de faire passer le point de terminaison de base de données de vos applications du cluster bleu au cluster vert.

Après le passage au numérique, la Blue/Green solution Neptune ne supprime pas l'ancien environnement de production bleu. Vous y aurez toujours accès pour une validation et des tests supplémentaires si nécessaire. Les frais de facturation standard s'appliqueront à ses instances jusqu'à ce que vous les supprimiez. La Blue/Green solution utilise également d'autres AWS services dont les coûts sont facturés aux prix normaux. Les détails relatifs à la suppression de la solution une fois que vous n'en avez plus besoin sont décrits dans la section sur le nettoyage.

Conditions préalables à l'utilisation de la pile Neptune Blue/Green

Avant de lancer la pile Neptune Blue/Green :

- Assurez-vous d'activer les flux Neptune dans le cluster de production (bleu).
- Toutes les instances du cluster bleu doivent présenter l'état disponible. Vous pouvez vérifier l'état des instances dans la console Neptune ou à l'aide de l'API describe-db-instances.
- Toutes les instances doivent également être synchronisées avec le groupe de paramètres du cluster de bases de données.
- La Blue/Green solution Neptune nécessite un point de terminaison DynamoDB VPC dans le VPC où se trouve votre cluster bleu. Consultez Utilisation des points de terminaison d'un VPC Amazon pour accéder à DynamoDB.

- Choisissez d'exécuter la solution à un moment où la charge de travail d'écriture sur le cluster de bases de données de production bleu sera aussi légère que possible. Évitez, par exemple, d'exécuter la solution lorsqu'un chargement en bloc doit avoir lieu ou lorsque vous prévoyez un grand nombre d'opérations d'écriture pour une autre raison.

À l'aide d'un CloudFormation modèle pour exécuter la solution Neptune Blue/Green

Vous pouvez l'utiliser AWS CloudFormation pour déployer la solution Neptune Blue/Green . Le CloudFormation modèle crée une instance Amazon EC2 dans le même VPC que votre base de données Neptune Blue Source, y installe la solution et l'exécute. Vous pouvez suivre sa progression dans des CloudWatch journaux, comme expliqué dans la section Surveillance de la progression.

Vous pouvez utiliser ces liens pour consulter le modèle de solution ou sélectionner le bouton Launch Stack pour le lancer dans la CloudFormation console :

Afficher	Afficher dans Designer	Launch Stack

Dans la console, choisissez la AWS région dans laquelle vous souhaitez exécuter la solution dans le menu déroulant en haut à droite de la fenêtre.

Définissez les paramètres de la pile comme suit :

- **`DeploymentID`**— Un identifiant unique pour chaque déploiement de Neptune Blue/Green .

  Il est utilisé comme identifiant de cluster de bases de données vert et comme préfixe pour nommer les nouvelles ressources créées lors du déploiement.
- **`NeptuneSourceClusterId`** : identifiant du cluster de bases de données bleu que vous souhaitez mettre à niveau.
- **`NeptuneTargetClusterVersion:`** : version du moteur Neptune vers laquelle vous souhaitez mettre à niveau le cluster de bases de données bleu.

  Cette valeur doit être plus récente que la version actuelle du moteur du cluster de bases de données bleu.

- **`DeploymentMode`** : indique s'il s'agit d'un nouveau déploiement ou d'une tentative de reprise d'un déploiement précédent. Lorsque vous utilisez un `DeploymentID` identique à celui d'un déploiement précédent, définissez `DeploymentMode` sur `resume`.

  Les valeurs valides sont `new` (par défaut) et `resume`.

- **`GraphQueryType`** : type de données de graphe correspondant à votre base de données.

  Les valeurs valides sont `propertygraph` (par défaut) et `rdf`.

- **`SubnetId`** : ID de sous-réseau provenant du même VPC que celui dans lequel se trouve le cluster de bases de données bleu (voir Connexion à un cluster de bases de données Neptune à partir d'une instance Amazon EC2 dans le même VPC).

  Fournissez l'ID d'un sous-réseau public si vous souhaitez accéder à l'instance par SSH via EC2 Connect.

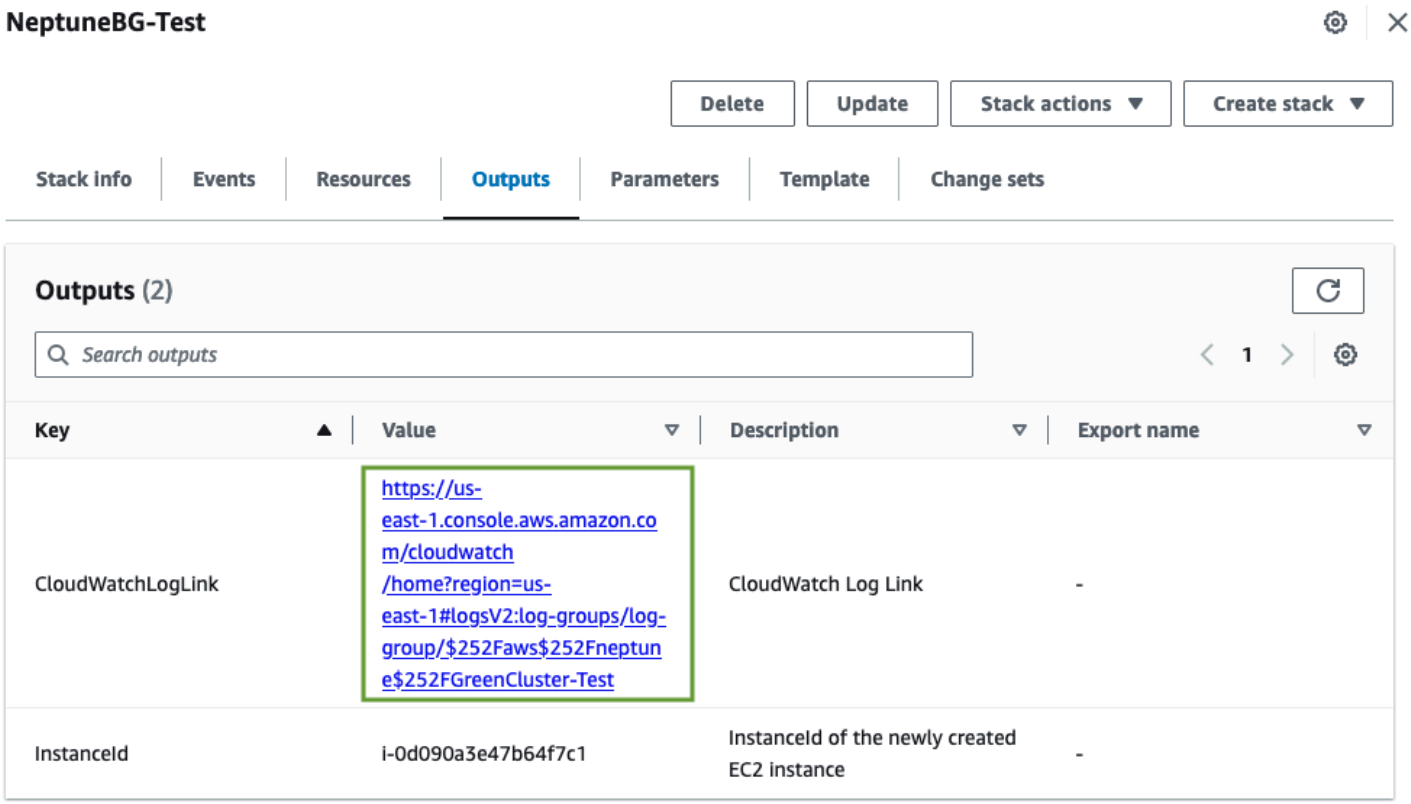
- **`InstanceSecurityGroup`** : groupe de sécurité de votre instance Amazon EC2.

  Le groupe de sécurité doit avoir accès à votre cluster de bases de données bleu, et vous devez être en mesure de vous connecter à l'instance par SSH. Consultez Créez un groupe de sécurité à l'aide de la console VPC..

Patientez jusqu'à ce que la pile soit terminée. Dès que cette étape est terminée, le lancement de la solution a lieu. Vous pouvez ensuite surveiller le processus de déploiement à l'aide CloudWatch des journaux, comme décrit dans la section suivante.

Surveillance de la progression d'un déploiement de Neptune Blue/Green

Vous pouvez suivre la progression de la Blue/Green solution Neptune en accédant à la CloudWatch console et en consultant les journaux du groupe de `/aws/neptune/`*`(Neptune Blue/Green deployment ID)`* CloudWatch journaux. Vous pouvez trouver un lien vers les CloudWatch journaux dans les sorties de la CloudFormation pile de la solution :



Si vous avez fourni un sous-réseau public en tant que paramètre de pile, vous pouvez également accéder par SSH à l'instance Amazon EC2 créée dans le cadre de la pile et vous référer au journal dans `/var/log/cloud-init-output.log`.

Le journal indique les actions entreprises par la Blue/Green solution Neptune, comme le montre cette capture d'écran :

```
============================================================

Neptune Blue Green Deployment Solution Version: 0.1.06012023

============================================================

Checking whether cluster with id = bg-06-01-14-20-29test-bg1-bgInt already exists.

BlueGreen deployment_mode = new

Didn't find any cluster with id bg-06-01-14-20-29test-bg1-bgInt

Cloned_cluster_id: bg-06-01-14-20-29test-bg1-bgInt

Replication_stack_name: bg-06-01-14-20-29test-bg1-bgInt-replication

DescribeDbClusters response for test-bg1-bgIntegTest-06-01-14-20-29: {'AllocatedStorage': 1,
'AvailabilityZones': ['us-east-1b', 'us-east-1c', 'us-east-1f'], 'BackupRetentionPeriod': 1,
'DBClusterIdentifier': 'test-bg1-bgintegtest-06-01-14-20-29', 'DBClusterParameterGroup': 'green-     -blue-
green-deployment-test-123456789012345-pg-tes710', 'DBSubnetGroup': 'default', 'Status': 'available',
'EarliestRestorableTime': datetime.datetime(2023, 6, 1, 8, 51, 23, 394000, tzinfo=tzlocal()), 'Endpoint':
'test-bg1-bgintegtest-06-01-14-20-29.cluster-critvszpmydm.us-east-1.neptune.amazonaws.com', 'ReaderEndpoint':
'test-bg1-bgintegtest-06-01-14-20-29.cluster-ro-critvszpmydm.us-east-1.neptune.amazonaws.com', 'MultiAZ':
False, 'Engine': 'neptune', 'EngineVersion': '1.2.0.0', 'LatestRestorableTime': datetime.datetime(2023, 6, 1,
8, 51, 23, 394000, tzinfo=tzlocal()), 'Port': 8182, 'MasterUsername': 'admin', 'PreferredBackupWindow':
'06:33-07:03', 'PreferredMaintenanceWindow': 'fri:09:44-fri:10:14', 'ReadReplicaIdentifiers': [],
'DBClusterMembers': [{'DBInstanceIdentifier': 'test-bg1-bgintegtest-06-01-14-20-29i-1', 'IsClusterWriter':
True, 'DBClusterParameterGroupStatus': 'in-sync', 'PromotionTier': 1}], 'VpcSecurityGroups':
```

Les messages du journal indiquent l'état de synchronisation entre les clusters bleu et vert :

```
DDB checkpoint {'S': '1'}, {'S': '6'}

DDB Checkpoint: {'checkpointSubSequenceNumber': {'S': '6'}, 'lastUpdateTime': {'N': '1685611142127'},
'leaseOwner': {'S': 'nobody'}, 'checkpoint': {'S': '1'}, 'leaseKey': {'S': 'bg-anl   -234567899-replication'}}

Time difference for last checkpoint and last stream event: 5841351

Stream eventId difference for last replication checkpoint and last stream event on the Source cluster: 0:0

Found region : us-east-1

Cloudwatch Log Url for blue green solution is https://us-east-1.console.aws.amazon.com/cloudwatch
/home?region=us-east-1#logsV2:log-groups/log-group/aws/neptune/bg

Cloudwatch dashboard url for replication is https://console.aws.amazon.com/cloudwatch/home?region=us-
east-1#dashboards:name=neptune-stream-poller-bg-an   -234567899-replication

Replication poller lambda arn is arn:aws:lambda:us-east-1:451235071234:function:bg-an   -234567899-replic-
NeptuneStreamPollerLambd-B6V1ytULgmSP. Look for CW log the poller lambda for more troubleshooting.

Stream Last EventId {'commitNum': 1, 'opNum': 6} on cluster : database-d61852469-t        -experiment.cluster-
critvszpmydm.us-east-1.neptune.amazonaws.com:8182

DDB checkpoint {'S': '1'}, {'S': '6'}

DDB Checkpoint: {'checkpointSubSequenceNumber': {'S': '6'}, 'lastUpdateTime': {'N': '1685611207245'},
'leaseOwner': {'S': 'nobody'}, 'checkpoint': {'S': '1'}, 'leaseKey': {'S': 'bg-ankig-234567899-replication'}}
```

Le processus de synchronisation vérifie le délai de réplication en calculant la différence entre le dernier flux du cluster bleu et `eventID` le point de contrôle de réplication présent dans la table de points de contrôle DynamoDB créée par la pile de réplication. Neptune-to-Neptune À l'aide de ces messages, vous pouvez surveiller la différence de réplication actuelle.

Passage du cluster bleu de production au cluster vert mis à jour

Avant de promouvoir le cluster vert en production, assurez-vous que la différence de validation entre les clusters bleu et vert est nulle, puis désactivez tout le trafic d'écriture vers le cluster bleu. Toute nouvelle écriture dans le cluster bleu lors du basculement du point de terminaison de la base de données vers le cluster vert entraînerait une corruption des données provoquée par l'écriture de données partielles dans les deux clusters. Vous n'avez pas encore besoin de désactiver le trafic de lecture.

Si vous avez activé l'authentification IAM sur le cluster source (bleu), veillez à mettre à jour toutes les politiques IAM utilisées dans vos applications pour qu'elles pointent vers le cluster vert (pour un exemple de politique de ce type, consultez cette stratégie d'accès illimité).

Après avoir désactivé le trafic d'écriture, attendez que la réplication soit terminée, puis activez le trafic d'écriture sur le cluster vert (mais pas sur le cluster bleu). Faites également passer le trafic de lecture du cluster bleu au cluster vert.

Nettoyage une fois la Blue/Green solution Neptune terminée

Après avoir promu le cluster intermédiaire (vert) en production, nettoyez les ressources créées par la solution Neptune Blue/Green :

- Supprimez l'instance Amazon EC2 créée pour exécuter la solution.
- Supprimez les CloudFormation modèles pour la réplication basée sur les flux Neptune qui maintenaient le cluster vert synchronisé avec le cluster bleu. Le modèle principal porte le nom de pile que vous avez fourni précédemment, et l'autre est composé de l'ID de déploiement suivi de "-replication", à savoir `(DeploymentID)-replication`.

La suppression CloudFormation de modèles ne supprime pas les clusters eux-mêmes. Une fois que vous avez vérifié que le cluster vert fonctionne comme prévu, vous pouvez choisir de créer un instantané avant de supprimer manuellement le cluster bleu.

Bonnes pratiques en matière de Blue/Green solutions Neptune

- Avant de passer le cluster vert en production, il est important de vérifier minutieusement qu'il fonctionne correctement. Vérifiez la cohérence des données et la configuration de la base de données. Il est possible que certaines des nouvelles versions du moteur nécessitent également des mises à niveau du client. Consultez les notes de mise à jour du moteur avant de procéder à la mise à niveau. Il vaut la peine de tester tout cela dans des environnements de développement, de test et de pré-production avant de commencer une blue/green mise à niveau en production.
- Il est préférable de passer du serveur bleu au serveur vert pendant la fenêtre de maintenance.
- Pour garantir que tout fonctionne correctement après la mise à niveau et la synchronisation, il est conseillé de conserver le cluster d'origine pendant un certain temps avant de le supprimer. Cela pourrait s'avérer utile en cas de problème imprévu.
- Évitez les opérations d'écriture lourdes telles que les chargements en masse lors de l'exécution de la Blue/Green solution Neptune, car elles peuvent entraîner un retard de réplication entraînant des temps d'arrêt importants. Idéalement, le délai entre la désactivation des écritures dans le cluster bleu et leur activation dans le cluster vert est de quelques instants seulement.

Résolution des problèmes liés à la solution Neptune Blue/Green

Les informations suivantes mettent en évidence les problèmes qui peuvent survenir au cours du processus de déploiement de la Blue/Green solution, tels que les conflits avec les clusters existants, la nécessité d'activer les flux Neptune, les opérations de chargement groupé en cours et les exigences de compatibilité des versions. En résolvant ces problèmes potentiels, vous pouvez garantir un déploiement fluide et réussi de la solution Neptune Blue/Green .

Erreurs signalées par la solution Neptune Blue/Green

- **`Cluster with id = (blue_green_deployment_id) already exists`**— Il existe un cluster avec identifiant`(blue_green_deployment_id)`.

  Fournissez un nouvel ID de déploiement ou définissez le mode de déploiement comme `resume` si le cluster a été créé lors d'une précédente exécution de Neptune Blue/Green .
- **`Streams should be enabled on the source Cluster for Blue Green Deployment`** : activez les flux Neptune sur le cluster bleu (source).
- **`No Bulkload should be in progress on source cluster: (cluster_id)`**— La Blue/Green solution Neptune prend fin si elle identifie un chargement en vrac en cours.

  Cela permet de s'assurer que le processus de synchronisation est capable de rattraper son retard par rapport aux écritures effectuées. Évitez ou annulez toute tâche de chargement en vrac en cours avant de démarrer la solution Neptune Blue/Green .
- **`Blue Green deployment requires instances to be in sync with db cluster parameter group`** : toute modification apportée au groupe de paramètres du cluster doit être synchronisée dans l'ensemble du cluster de bases de données. Consultez Groupes de paramètres Amazon Neptune.
- **`Invalid target engine version for Blue Green Deployment`** : la version cible du moteur doit être répertoriée comme active dans Versions du moteur pour Amazon Neptune et doit être supérieure à la version actuelle du moteur du cluster source (bleu).

Création d'un utilisateur IAM avec les autorisations nécessaires pour accéder à Neptune

Pour accéder à la console Neptune afin de créer et de gérer un cluster de bases de données Neptune, vous devez créer un utilisateur IAM doté de toutes les autorisations nécessaires.

La première étape consiste à créer une politique de rôle liée à un service pour Neptune :

Création d'une politique de rôle liée à un service pour Amazon Neptune

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Dans le panneau de navigation de gauche, sélectionnez Policies (Politiques).
3. Sur la page Politiques, sélectionnez Créer une politique.
4. Sur la page Créer une politique, sélectionnez l'onglet JSON et copiez la politique de rôle suivante liée à un service :

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
      "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"rds.amazonaws.com"
        }
      }
    }
  ]
}
```

5. Sélectionnez Suivant : Balises, puis sur la page Ajouter des balises, sélectionnez Suivant : Vérification.
6. Sur la page Réviser la politique, nommez la nouvelle politique « NeptuneServiceLinked ».

Pour plus d'informations sur les rôles liés à un service, consultez Utilisation de rôles liés à un service pour Amazon Neptune.

Création d'un utilisateur IAM avec toutes les autorisations nécessaires

Créez ensuite l'utilisateur IAM avec les politiques gérées appropriées associées qui accorderont les autorisations dont vous avez besoin, ainsi que la politique de rôle liée à un service que vous avez créée (nommée ici `NeptuneServiceLinked`) :

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Dans le panneau de navigation de gauche, sélectionnez Utilisateurs, puis sur la page Utilisateurs, choisissez Ajouter des utilisateurs.
3. Sur la page Ajouter un utilisateur, entrez le nom du nouvel utilisateur IAM, choisissez Clé d'accès - Accès programmatique pour le type AWS d'identifiant, puis choisissez Suivant : Autorisations.
4. Sur la page Définir les autorisations, dans la zone Politiques de filtre, tapez « Neptune ». Sélectionnez maintenant les politiques suivantes parmi celles qui sont répertoriées :

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked(en supposant que c'est ainsi que vous avez nommé la politique de rôle liée au service que vous avez créée précédemment).
5. Tapez ensuite « VPC » dans le champ Politiques de filtre à la place de « Neptune ». Sélectionnez Amazon VPCFull Access dans les politiques répertoriées.
6. Sélectionnez Suivant : Balises, puis sur la page Ajouter des balises, sélectionnez Suivant : Vérification.
7. Sur la page Vérification, vérifiez que toutes les politiques suivantes sont désormais associées au nouvel utilisateur :

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked
   - Amazon VPCFull Access

   Sélectionnez ensuite Créer un utilisateur.

8. Enfin, téléchargez et enregistrez l'ID de clé d'accès et la clé d'accès secrète du nouvel utilisateur.

Pour interagir avec d'autres services tels qu'Amazon Simple Storage Service (Amazon S3), vous devez ajouter des autorisations et des relations d'approbation supplémentaires.

Groupes de paramètres Amazon Neptune

Vous gérez votre configuration de base de données dans Amazon Neptune à l'aide de paramètres dans un groupe de paramètres. Les groupes de paramètres servent de conteneurs pour les valeurs de configuration de moteur qui sont appliquées à une ou plusieurs instances de base de données.

Il existe deux types de groupes de paramètres : les groupes de paramètres de cluster de bases de données et les groupes de paramètres de base de données.

- Les groupes de paramètres de base de données s'appliquent au niveau de l'instance et sont généralement associés aux paramètres du moteur de graphe Neptune, comme le paramètre `neptune_query_timeout`.
- Les groupes de paramètres de cluster de base de données s'appliquent à chaque instance du cluster et ont généralement des valeurs plus générales. Chaque cluster Neptune est associé à un groupe de paramètres de cluster de bases de données. Toutes les instances de base de données au sein de ce cluster héritent des valeurs de configuration du moteur contenues dans le groupe de paramètres de cluster de bases de données.

Les valeurs de configuration que vous modifiez dans le groupe de paramètres de cluster de base de données remplacent les valeurs par défaut du groupe de paramètres de base de données. Si vous modifiez les valeurs correspondantes dans le groupe de paramètres de base de données, ces valeurs remplacent celles du groupe de paramètres de cluster de bases de données.

Un groupe de paramètres de base de données par défaut est utilisé si vous créez une instance de base de données sans spécifier un groupe de paramètres de base de données personnalisé. Vous ne pouvez pas modifier les valeurs de paramètre d'un groupe de paramètres de base de données par défaut. Pour modifier les paramètres par défaut, vous devez créer un groupe de paramètres de base de données. Il n'est pas possible de modifier tous les paramètres de moteur de base de données dans un groupe de paramètres de base de données que vous créez.

Les groupes de paramètres sont créés dans des familles compatibles avec des versions spécifiques du moteur Neptune. Lorsque vous effectuez une mise à niveau vers une nouvelle version majeure ou secondaire du moteur, vous devrez peut-être recréer vos groupes de paramètres personnalisés en utilisant la famille de groupes de paramètres correspondante pour cette version.

La dénomination de la famille de groupes de paramètres suit le modèle`neptuneX.Y`, où `X.Y` correspond à la version du moteur. Par exemple :

- `neptune1`— pour les versions du moteur antérieures à 1.2.0.0
- `neptune1.2`— pour les versions de moteur 1.2.x
- `neptune1.3`— pour les versions de moteur 1.3.x
- `neptune1.4`— pour les versions de moteur 1.4.x

Lors de la mise à niveau de votre cluster Neptune, consultez les notes de publication de la version de votre moteur cible pour déterminer si une nouvelle famille de groupes de paramètres est requise. Dans ce cas, vous devez recréer tous les groupes de paramètres personnalisés de la nouvelle famille avant de procéder à la mise à niveau.

Certains paramètres Neptune sont statiques, tandis que d'autres sont dynamiques. Les différences sont les suivantes :

Paramètres statiques

- Un paramètre statique est un paramètre qui ne prend effet qu'après le redémarrage d'une instance de base de données. En d'autres termes, lorsque vous modifiez un paramètre statique et que vous enregistrez le groupe de paramètres de base de données d'une instance, vous devez redémarrer manuellement l'instance de base de données pour que la modification du paramètre soit appliquée. Actuellement, tous les paramètres au niveau des instances Neptune (dans un groupe de paramètres de base de données plutôt que dans un groupe de paramètres de cluster de bases de données) sont statiques.
- Lorsque vous modifiez un paramètre statique au niveau du cluster et que vous enregistrez le groupe de paramètres de base de données d'un cluster, la modification de ce paramètre est appliquée après que vous avez redémarré manuellement chaque instance de base de données du cluster.

Paramètres dynamiques

- Un paramètre dynamique est un paramètre qui prend effet presque immédiatement après sa mise à jour dans son groupe de paramètres. En d'autres termes, il n'est pas nécessaire de redémarrer une instance de base de données après avoir mis à jour un paramètre dynamique pour que la modification prenne effet.
- Attendez-vous à un léger délai avant que la modification d'un paramètre de cluster dynamique soit appliquée à toutes les instances de base de données.

- Une valeur de paramètre dynamique mise à jour n'est pas appliquée aux demandes en cours d'exécution, mais uniquement à celles soumises après la modification.
- Lorsque vous modifiez un paramètre dynamique au niveau du cluster, par défaut, la modification de ce paramètre est appliquée à votre cluster de bases de données immédiatement, sans nécessiter de redémarrage. Pour différer la modification des paramètres jusqu'au redémarrage des instances de base de données du cluster, vous pouvez utiliser le pour définir la AWS CLI valeur sur `ApplyMethod` `pending-reboot` pour la modification des paramètres.

Actuellement, tous les paramètres sont statiques, à l'exception des nouveaux paramètres de cluster suivants :

- `neptune_enable_slow_query_log` (au niveau du cluster)
- `neptune_slow_query_log_threshold` (au niveau du cluster)

Voici quelques éléments importants que vous devez connaître concernant l'utilisation de paramètres dans un groupe de paramètres DB :

- La configuration incorrecte de paramètres dans un groupe de paramètres DB peut avoir des effets contraires involontaires, dont une dégradation de la performance et une instabilité du système. Montrez-vous toujours prudent lorsque vous modifiez des paramètres de base de données et sauvegardez vos données avant de modifier un groupe de paramètres de base de données. Testez les modifications de paramètres de votre groupe de paramètres sur une instance de base de données test avant d'appliquer ces modifications à une instance de base de données de production.
- Lorsque vous modifiez le groupe de paramètres DB associé à une instance de base de données, vous devez redémarrer manuellement l'instance avant que le nouveau groupe de paramètres DB soit utilisé par l'instance de base de données.

> **Note**
>
> Avant la Sortie : 1.2.0.0 (21/07/2022), toutes les instances de réplica en lecture d'un cluster de bases de données étaient automatiquement redémarrées lors du redémarrage de l'instance (d'enregistreur) principale.
> À compter de la Sortie : 1.2.0.0 (21/07/2022), le redémarrage de l'instance principale n'entraîne le redémarrage d'aucune des instances de réplica. Dès lors, si vous modifiez un

paramètre au niveau du cluster, vous devez redémarrer chaque instance séparément pour que cette modification de paramètre s'applique.

Modification d'un groupe de paramètres de cluster de base de données ou d'un groupe de paramètres de base de données

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Choisissez Parameter groups (Groupes de paramètres) dans le volet de navigation.
3. Choisissez le lien Name (Nom) pour le groupe de paramètres de base de données à modifier.

   (Facultatif) Choisissez Create parameter group (Créer un groupe de paramètres) pour créer un nouveau groupe de paramètres de cluster. Choisissez ensuite le Name (Nom) du nouveau groupe de paramètres.

   > **Important**
   >
   > Cette étape est obligatoire si vous ne disposez que du groupe de paramètres de cluster de base de données par défaut, car ce groupe de paramètres ne peut pas être modifié.

4. Recherchez le paramètre et cliquez sur le champ Valeur à côté de la colonne Nom.
5. Entrez la valeur autorisée et cochez la case à côté du champ de valeur.
6. Sélectionnez Enregistrer les modifications.
7. Redémarrez chaque instance de base de données du cluster Neptune si vous modifiez un paramètre de cluster de bases de données, ou une ou plusieurs instances spécifiques si vous modifiez un paramètre d'instance de base de données.

Création d'un groupe de paramètres de base de données ou d'un groupe de paramètres de cluster de bases de données

Vous pouvez facilement utiliser la console Neptune pour créer un groupe de paramètres :

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.

2. Choisissez Parameter groups (Groupes de paramètres) dans le volet de navigation de gauche.
3. Choisissez Create DB parameter group (Créer un groupe de paramètres de base de données).

   La page Create DB parameter group (Créer un groupe de paramètres de base de données) s'affiche.
4. Dans la liste des familles de groupes de paramètres, choisissez la famille qui correspond à la version de votre moteur Neptune cible (par exemple, neptune1.2, neptune1.3 ou neptune1.4).
5. Dans la liste Type, choisissez DB Parameter Group (Groupe de paramètres de base de données) ou DB Cluster Parameter Group (Groupe de paramètres de cluster de base de donnée).
6. Dans la zone Nom du groupe, saisissez le nom du nouveau groupe de paramètres de base de données.
7. Dans la zone Description, saisissez une description pour le nouveau groupe de paramètres de base de données.
8. Choisissez Créer.

Vous pouvez également créer un groupe de paramètres à l'aide de l' AWS CLI :

```
aws neptune create-db-parameter-group \
  --db-parameter-group-name (a name for the new DB parameter group) \
  --db-parameter-group-family (the family matching your engine version, such as
 neptune1.2, neptune1.3, or neptune1.4) \
  --description (a description for the new DB parameter group)
```

Paramètres Amazon Neptune

Vous gérez votre configuration de base de données dans Amazon Neptune à l'aide de paramètres dans des groupes de paramètres. Les paramètres suivants sont disponibles pour configurer votre base de données Neptune :

Paramètres de niveau cluster

- neptune_enable_audit_log
- neptune_enable_slow_query_log
- neptune_slow_query_log_threshold
- neptune_lab_mode
- neptune_query_timeout
- neptune_streams
- neptune_streams_expiry_days
- neptune_lookup_cache
- neptune_autoscaling_config
- neptune_ml_iam_role
- neptune_ml_endpoint
- neptune_enable_inline_server_generated_edge_id

Paramètres de niveau instance

- neptune_dfe_query_engine
- neptune_query_timeout
- neptune_result_cache
- UndoLogPurgeConfig

Paramètres obsolètes

- neptune_enforce_ssl

`neptune_enable_audit_log` (paramètre de niveau cluster)

Ce paramètre active la journalisation des audits pour Neptune.

Les valeurs autorisées sont `0` (désactivé) et `1` (activé). La valeur par défaut est `0`.

Ce paramètre est statique, ce qui signifie que les modifications apportées ne prennent effet sur aucune instance tant qu'elle n'a pas été redémarrée.

Vous pouvez publier des journaux d'audit sur Amazon CloudWatch, comme décrit dansUtilisation de la CLI pour publier les journaux d'audit de Neptune dans Logs CloudWatch.

`neptune_enable_slow_query_log` (paramètre de niveau cluster)

Utilisez ce paramètre pour activer ou désactiver la fonctionnalité de journalisation des requêtes lentes de Neptune.

Il s'agit d'un paramètre dynamique. Autrement dit, la modification de sa valeur ne nécessite ni ne provoque le redémarrage du cluster de bases de données.

Les valeurs autorisées sont les suivantes :

- **`info`** : active la journalisation des requêtes lentes et enregistre les attributs sélectionnés susceptibles de contribuer au ralentissement des performances.
- **`debug`** : active la journalisation des requêtes lentes et enregistre tous les attributs disponibles de la requête exécutée.
- **`disabled`** : désactive la journalisation des requêtes lentes.

La valeur par défaut est `disabled`.

Vous pouvez publier des journaux de requêtes lentes sur Amazon CloudWatch, comme décrit dans. Utilisation de la CLI pour publier les journaux de requêtes lentes de Neptune dans Logs CloudWatch

`neptune_slow_query_log_threshold` (paramètre de niveau cluster)

Ce paramètre indique le délai d'exécution, en millisecondes, après lequel une requête est considérée comme une requête lente. Si la journalisation des requêtes lentes est activée, les requêtes dont l'exécution dépasse ce seuil sont journalisées avec certains de leurs attributs.

La valeur par défaut est de 5 000 millisecondes (5 secondes).

Il s'agit d'un paramètre dynamique. Autrement dit, la modification de sa valeur ne nécessite ni ne provoque le redémarrage du cluster de bases de données.

`neptune_lab_mode` (paramètre de niveau cluster)

Lorsqu'il est défini, ce paramètre active des fonctionnalités expérimentales spécifiques de Neptune. Consultez Mode expérimental Neptune pour connaître les fonctions expérimentales actuellement disponibles.

Ce paramètre est statique, ce qui signifie que les modifications apportées ne prennent effet sur aucune instance tant qu'elle n'a pas été redémarrée.

Les fonctionnalités du mode expérimental sont généralement désactivées par défaut. Pour activer ou désactiver explicitement une fonctionnalité expérimentale, incluez *`(feature name)`* `=enabled` ou *`(feature name)`* `=disabled` dans ce paramètre. Vous pouvez activer ou désactiver plusieurs fonctions en les séparant par des virgules, comme ceci :

`(feature #1 name)``=enabled,` *`(feature #2 name)`*`=enabled`

`neptune_query_timeout` (paramètre de niveau cluster)

Spécifie une durée d'expiration particulière pour les requêtes de graphe, en millisecondes.

Les valeurs autorisées vont de `10` à `2,147,483,647` (2^{31} - 1). La valeur par défaut est `120,000` (2 minutes).

Ce paramètre est statique, ce qui signifie que les modifications apportées ne prennent effet sur aucune instance tant qu'elle n'a pas été redémarrée.

Lorsque plusieurs paramètres de délai d'attente sont configurés (au niveau du cluster, au niveau de l'instance et par requête), le tableau suivant indique quelle valeur de délai est prioritaire :

PG de clusters	Instance PG	Indicateur de requête	Outcome
Par défaut	Par défaut	Aucune	Cluster
Personnalisé	Par défaut	Aucune	Cluster
Personnalisé	Personnalisé	Aucune	Instance
Par défaut	Personnalisé	Aucune	Instance

PG de clusters	Instance PG	Indicateur de requête	Outcome
N'importe lequel	N'importe lequel	le plus bas	Query
Par défaut	Personnalisé	pas le plus bas	Instance
Personnalisé	Par défaut	pas le plus bas	Cluster
Personnalisé	Personnalisé	pas le plus bas	Instance

> **Note**
>
> Il est possible que vous encouriez des coûts inattendus si vous définissez une valeur trop élevée pour le délai d'expiration des requêtes, en particulier sur une instance sans serveur. En l'absence d'un délai raisonnable d'expiration des requêtes, vous risquez de lancer par inadvertance une requête qui s'exécute bien plus longtemps que prévu, entraînant ainsi des coûts que vous n'aviez pas anticipés. Cela est particulièrement vrai pour une instance sans serveur, qui pourrait passer à un type d'instance volumineux et coûteux lors de l'exécution de la requête.
>
> Pour éviter des dépenses imprévues de ce type, utilisez une valeur de délai d'expiration qui convient à la plupart des requêtes et qui n'implique que l'expiration de celles dont le délai est étonnamment long.

`neptune_streams` (paramètre de niveau cluster)

Active ou désactive Flux Neptune.

Ce paramètre est statique, ce qui signifie que les modifications apportées ne prennent effet sur aucune instance tant qu'elle n'a pas été redémarrée.

Les valeurs autorisées sont `0` (désactivé, qui est la valeur par défaut) et 1 (activé).

`neptune_streams_expiry_days` (paramètre de niveau cluster)

Spécifie le nombre de jours qui s'écoulent avant que le serveur ne supprime les enregistrements de flux.

Les valeurs autorisées vont de 1 à `90` inclus. La valeur par défaut est 7.

Ce paramètre a été introduit dans la version 1.2.0.0 du moteur.

Ce paramètre est statique, ce qui signifie que les modifications apportées ne prennent effet sur aucune instance tant qu'elle n'a pas été redémarrée.

`neptune_lookup_cache` (paramètre de niveau cluster)

Désactive ou réactive le cache de recherche Neptune sur les instances `R5d`.

Ce paramètre est statique, ce qui signifie que les modifications apportées ne prennent effet sur aucune instance tant qu'elle n'a pas été redémarrée.

Les valeurs autorisées sont `1` (activé) et `0` (désactivé). La valeur par défaut est `0`, mais chaque fois qu'une instance `R5d` est créée dans le cluster de bases de données, le paramètre `neptune_lookup_cache` est automatiquement défini sur `1`, et un cache de recherche est créé sur cette instance.

`neptune_autoscaling_config` (paramètre de niveau cluster)

Définit les paramètres de configuration des instances de réplica en lecture créées et gérées par l'auto-scaling Neptune.

Ce paramètre est statique, ce qui signifie que les modifications apportées ne prennent effet sur aucune instance tant qu'elle n'a pas été redémarrée.

À l'aide d'une chaîne JSON que vous définissez comme valeur du paramètre `neptune_autoscaling_config`, vous pouvez spécifier :

- Type d'instance utilisé par l'auto-scaling Neptune pour toutes les nouvelles instances de réplica en lecture qu'il crée.
- Les fenêtres de maintenance attribuées à ces réplicas en lecture.
- Les balises à associer à tous les nouveaux réplicas en lecture.

La chaîne JSON a une structure comme celle-ci :

```
"{
  \"tags\": [
    { \"key\" : \"reader tag-0 key\", \"value\" : \"reader tag-0 value\" },
    { \"key\" : \"reader tag-1 key\", \"value\" : \"reader tag-1 value\" },
  ],
```

```
  \"maintenanceWindow\" : \"wed:12:03-wed:12:33\",
  \"dbInstanceClass\" : \"db.r5.xlarge\"
}"
```

Notez que les guillemets de la chaîne doivent tous être précédés d'une barre oblique inverse (\).

Les trois paramètres de configuration non spécifiés dans le paramètre `neptune_autoscaling_config` sont copiés à partir de la configuration de l'instance d'enregistreur principale du cluster de bases de données.

`neptune_ml_iam_role` (paramètre de niveau cluster)

Spécifie l'ARN du rôle IAM utilisé dans Neptune ML. Cette valeur peut être n'importe quel ARN de rôle IAM valide.

Ce paramètre est statique, ce qui signifie que les modifications apportées ne prennent effet sur aucune instance tant qu'elle n'a pas été redémarrée.

Vous pouvez spécifier l'ARN du rôle IAM par défaut pour le machine learning au niveau des graphes.

`neptune_ml_endpoint` (paramètre de niveau cluster)

Spécifie le point de terminaison utilisé pour Neptune ML. La valeur peut être n'importe quel nom de point de terminaison SageMaker AI valide.

Ce paramètre est statique, ce qui signifie que les modifications apportées ne prennent effet sur aucune instance tant qu'elle n'a pas été redémarrée.

Vous pouvez spécifier le point de terminaison d' SageMaker IA par défaut pour l'apprentissage automatique sur des graphiques.

`neptune_enable_inline_server_generated_edge_id` (paramètre de niveau cluster)

Activez ou désactivez la fonctionnalité Edge ID générée par le serveur en ligne Neptune.

Ce paramètre est statique, ce qui signifie que les modifications apportées ne prennent effet sur aucune instance tant qu'elle n'a pas été redémarrée.

Les valeurs autorisées sont `1` (activé) et `0` (désactivé). La valeur par défaut est `0`.

`neptune_dfe_query_engine` (paramètres de niveau instance)

Ce paramètre d'instance de base de données contrôle la manière dont le moteur de requête DFE est utilisé. Les valeurs autorisées sont les suivantes :

Ce paramètre est statique, ce qui signifie que les modifications apportées ne prennent effet sur aucune instance tant qu'elle n'a pas été redémarrée.

- **`enabled`** : fait en sorte que le moteur DFE soit utilisé dans la mesure du possible, sauf lorsque l'indicateur de requête `useDFE` est présent et défini sur `false`.
- **`viaQueryHint`** (valeur par défaut) : fait en sorte que le moteur DFE soit utilisé uniquement pour les requêtes qui incluent explicitement l'indicateur de requête `useDFE` défini sur `true`.

Si ce paramètre n'a pas été défini explicitement, la valeur par défaut, `viaQueryHint`, est utilisée au démarrage de l'instance.

> **Note**
>
> Toutes les requêtes openCypher sont exécutées par le moteur DFE, quelle que soit la manière dont ce paramètre est défini.

`neptune_query_timeout` (paramètres de niveau instance)

Ce paramètre d'instance de base de données spécifie un délai d'expiration pour les requêtes de graphe, en millisecondes, pour une instance.

Ce paramètre est statique, ce qui signifie que les modifications apportées ne prennent effet sur aucune instance tant qu'elle n'a pas été redémarrée.

Les valeurs autorisées vont de `10` à `2,147,483,647` (2^{31} - 1). La valeur par défaut est `120,000` (2 minutes).

> **Note**
>
> Il est possible que vous encouriez des coûts inattendus si vous définissez une valeur trop élevée pour le délai d'expiration des requêtes, en particulier sur une instance sans serveur. En l'absence d'un délai raisonnable d'expiration des requêtes, vous risquez de lancer par

inadvertance une requête qui s'exécute bien plus longtemps que prévu, entraînant ainsi des coûts que vous n'aviez pas anticipés. Cela est particulièrement vrai pour une instance sans serveur, qui pourrait passer à un type d'instance volumineux et coûteux lors de l'exécution de la requête.
Pour éviter des dépenses imprévues de ce type, utilisez une valeur de délai d'expiration qui convient à la plupart des requêtes et qui n'implique que l'expiration de celles dont le délai est étonnamment long.

neptune_result_cache (paramètres de niveau instance)

neptune_result_cache : ce paramètre d'instance de base de données active ou désactive Mise en cache des résultats de requête.

Ce paramètre est statique, ce qui signifie que les modifications apportées ne prennent effet sur aucune instance tant qu'elle n'a pas été redémarrée.

Les valeurs autorisées sont `0` (désactivé, qui est la valeur par défaut) et 1 (activé).

UndoLogPurgeConfig (paramètres de niveau instance)

UndoLogPurgeConfig— Utilisez ce paramètre pour activer ou désactiver la UndoLog purge agressive dans Neptune.

Les valeurs autorisées sont `default` les suivantes : qui utilise le nombre standard de threads pour la purge des journaux d'annulation et `aggressive` qui utilise un nombre accru de threads pour accélérer le nettoyage des journaux d'annulation. Lorsque l'`agressive`option est sélectionnée, vous pouvez vous attendre à observer une valeur plus élevée pour la `NumUndoPagesPurged` métrique.

neptune_enforce_ssl (paramètre au niveau du cluster OBSOLÈTE)

(Obsolète) Certaines régions autorisaient les connexions HTTP à Neptune, et ce paramètre était utilisé pour forcer toutes les connexions à utiliser le protocole HTTPS lorsqu'il était défini sur 1.
Ce paramètre n'est toutefois plus pertinent, car Neptune n'accepte désormais que les connexions HTTPS dans toutes les régions.

Lancement d'un cluster de base de données Neptune à l'aide du AWS Management Console

Le moyen le plus simple de lancer un nouveau cluster de base de données Neptune consiste à utiliser un CloudFormation modèle qui crée toutes les ressources nécessaires pour vous, comme expliqué dans. Créer un cluster Neptune

Si vous préférez, vous pouvez également utiliser la console Neptune pour lancer un nouveau cluster de bases de données manuellement, comme expliqué ici.

> **Note**
>
> Avant de pouvoir accéder à la console Neptune pour créer un cluster Neptune, vous devez disposer d'un utilisateur disposant des autorisations suffisantes. Si votre utilisateur actuel ne dispose pas de ces autorisations, vous pouvez créer un utilisateur IAM doté des autorisations nécessaires, comme expliqué dansCréation d'un utilisateur IAM avec les autorisations nécessaires pour accéder à Neptune.

Une fois que vous avez vérifié que votre utilisateur dispose des autorisations appropriées, ou que vous avez créé un utilisateur avec les autorisations appropriées, connectez-vous en AWS Management Console tant qu'utilisateur IAM et suivez les étapes ci-dessous pour créer un nouveau cluster de base de données :

Pour lancer un cluster de bases de données Neptune à l'aide de la console

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Accédez à la page Clusters sous Bases de données et choisissez Créer une base de données, ce qui ouvre la page Créer une base de données.
3. Sous Paramètres, entrez le nom de votre nouveau cluster de bases de données ou acceptez le nom par défaut qui y est fourni. Ce nom est utilisé dans l'adresse de point de terminaison de l'instance et doit satisfaire aux contraintes suivantes :
   - Il doit contenir entre 1 et 63 caractères alphanumériques ou traits d'union.
   - Son premier caractère doit être une lettre.
   - Il ne peut pas se terminer par un trait d'union ou contenir deux traits d'union consécutifs.

- Il doit être unique pour toutes les instances de base de données de votre AWS compte dans une AWS région donnée.

4. Sous Modèles, choisissez Production ou Développement et test.
5. Sous Taille de l'instance debase de données, choisissez une taille d'instance. Cela détermine la capacité de traitement et de mémoire de l'instance d'enregistreur principale du nouveau cluster de bases de données.

   Si vous avez sélectionné le modèle Production, vous avez uniquement le choix entre les classes optimisées pour la mémoire disponibles dans la liste, mais si vous avez sélectionné Développement et test, vous pouvez également choisir des classes extensibles plus économiques (voir Instances extensibles T3 pour une discussion sur les classes extensibles).

   > **Note**
   >
   > Neptune ne prend plus en charge les types d'R4instances.

6. Sous Disponibilité et durabilité, vous pouvez choisir d'activer ou non le déploiement multi-availability-zone (multi-AZ). Le modèle de production permet le déploiement multi-AZ par défaut, contrairement au modèle de développement et de test. Si le déploiement multi-AZ est activé, Neptune localise les instances de réplication en lecture que vous créez dans différentes zones AZs de disponibilité () afin d'améliorer la disponibilité.
7. Sous Connectivité, sélectionnez parmi les choix disponibles le cloud privé virtuel (VPC) qui hébergera le nouveau cluster de bases de données. Ici, vous pouvez choisir Créer un VPC si vous souhaitez que Neptune crée le VPC pour vous. Vous devez créer une instance Amazon EC2 dans ce même VPC pour accéder à l'instance Neptune (pour plus d'informations, consultez Sécurisation de votre base de données Amazon Neptune avec Amazon VPC). Notez que vous ne pourrez pas modifier le VPC après la création du cluster de bases de données.

   Si nécessaire, vous pouvez configurer davantage la connectivité du cluster sous Configuration de connectivité supplémentaire :

   a. Sous Groupe de sous-réseaux, vous pouvez choisir le groupe de sous-réseaux de base de données Neptune à utiliser pour le nouveau cluster de bases de données. Si votre VPC ne comporte pas encore de groupes de sous-réseaux, Neptune vous créera un groupe de sous-réseaux de base de données (voir Sécurisation de votre base de données Amazon Neptune avec Amazon VPC).

b. Sous Groupes de sécurité VPC, choisissez un ou plusieurs groupes de sécurité VPC existants pour sécuriser l'accès réseau au nouveau cluster de bases de données, ou choisissez Créer si vous souhaitez que Neptune en crée un pour vous, puis fournissez un nom pour le nouveau groupe de sécurité VPC (voir Créez un groupe de sécurité à l'aide de la console VPC.).

c. Sous Port de base de données, entrez le TCP/IP port que la base de données utilisera pour les connexions aux applications. Neptune utilise le numéro de port 8182 par défaut.

8. Sous Configuration du bloc-notes, choisissez Créer un bloc-notes si vous souhaitez que Neptune vous crée des blocs-notes Jupyter dans le workbench Neptune (voir Utilisation d'Amazon Neptune avec des carnets graphiques et Utilisation du workbench Neptune pour héberger des blocs-notes Neptune). Vous pouvez ensuite choisir la manière dont les nouveaux blocs-notes doivent être configurés :

   a. Sous Type d'instance de bloc-notes, choisissez parmi les classes d'instances disponibles pour votre bloc-notes.

   b. Dans Nom du bloc-notes, entrez un nom pour le bloc-notes.

   c. Si vous le souhaitez, vous pouvez également saisir une description du bloc-notes sous Description (facultatif).

   d. Sous Nom du rôle IAM, choisissez soit de demander à Neptune de créer un rôle IAM pour le bloc-notes et d'entrer un nom pour le nouveau rôle, soit de sélectionner un rôle IAM existant parmi les rôles disponibles.

   e. Enfin, choisissez si votre bloc-notes se connecte à Internet directement, via Amazon SageMaker AI ou via un VPC doté d'une passerelle NAT. Consultez Connexion d'une instance de bloc-notes aux ressources d'un VPC pour plus d'informations.

9. Sous Balises, vous pouvez associer jusqu'à 50 balises à votre nouveau cluster de bases de données.

10. Sous Configuration supplémentaire, vous pouvez définir d'autres paramètres pour votre nouveau cluster de bases de données (dans de nombreux cas, vous pouvez les ignorer et accepter les valeurs par défaut pour le moment) :

Option	Actions possibles
Identifiant d'instance de base de données	Vous pouvez fournir un nom pour l'instanc e d'enregistreur du cluster. Si vous n'en fournissez pas, un identifiant par défaut basé

Option	Actions possibles
	sur le nom du cluster sera utilisé. Si c'est le cas, spécifiez un nom unique pour toutes les instances de base de données détenues par votre AWS compte dans la région actuelle. L'identifiant d'instance de base de données n'est pas sensible à la casse, mais il est stocké intégralement en minuscules.
Groupe de paramètres de cluster de bases de données	Sélectionnez un groupe de paramètres de cluster de bases de données pour définir la configuration par défaut pour l'ensembl e des instances de base de données du cluster. Sauf indication contraire de votre part, Neptune utilise un groupe de paramètre s de cluster de bases de données par défaut. Pour plus d'informations sur les groupes de paramètres, consultez Groupes de paramètre s Amazon Neptune.
Groupe de paramètres de base de données	Sélectionnez un groupe de paramètres de base de données pour définir la configura tion de l'instance de base de données principale dans le cluster. Sauf indication contraire de votre part, Neptune utilisera un groupe de paramètres par défaut. Pour plus d'informations sur les groupes de paramètre s, consultez Groupes de paramètres.

Option	Actions possibles
Authentification de base de données IAM	Si vous cochez Activer l'authentification IAM DB, tous les accès à votre base de données seront authentifiés à l'aide de Gestion des identités et des accès AWS (IAM). **Important** Cela nécessite que vous signiez toutes les demandes avec AWS la signature Signature Version 4. Pour de plus amples informations, veuillez consulter Authentification de votre base de données Amazon Neptune avec Gestion des identités et des accès AWS.
Priorité de basculement	Choisissez `No preference` ou un niveau de priorité pour le basculement. Si vous choisissez un niveau dans lequel il existe un conflit, le réplica ayant la même taille que l'instance principale est sélectionné.
Période de rétention des sauvegardes	Sélectionnez la durée, comprise entre 1 et 35 jours, pendant laquelle Neptune doit conserver les sauvegardes automatiques de cette instance de base de données. Vous ne pouvez effectuer une point-in-time restauration (PITR) que pendant la période de conservation des sauvegardes.
Copier les balises aux instantanés	(Activée par défaut) Cette option permet de copier toutes les balises associées à votre cluster de bases de données sur tous ses instantanés.

Option	Actions possibles
Activer le chiffrement	(Activée par défaut) Cette option permet de chiffrer les données au repos de votre cluster de bases de données. Dans ce cas, choisissez la clé principale utilisée pour protéger la clé permettant de chiffrer ce volume de base de données. Vous pouvez sélectionner les clés `aws/rds` par défaut, choisir des clés principales dans votre compte ou entrer l'ARN d'une clé issue d'un autre compte. Vous pouvez créer une clé de chiffrement principale dans l'onglet Clés de chiffrement de la console IAM. Pour de plus amples informations, veuillez consulter Chiffrer les ressources Neptune au repos.
Journal d'audit	Cochez cette case si vous souhaitez que les journaux d'audit de votre cluster de base de données soient publiés dans CloudWatch Logs.
Enable auto minor version upgrade (Activer la mise à niveau automatique de versions mineures)	(Activée par défaut) Cette option entraîne la mise à niveau automatique de votre cluster de bases de données vers de nouvelles versions mineures du moteur après leur publication. Les mises à niveau automatiq ues se produisent pendant la fenêtre de maintenance de la base de données. Consultez En utilisant AutoMinorVersionUp grade.

Option	Actions possibles
Fenêtre de maintenance	Vous pouvez sélectionner une période spécifique pendant laquelle vous souhaitez que les modifications en attente soient apportées à votre cluster de bases de données, telles qu'une modification d'une classe d'instances de base de données ou un correctif automatique du moteur. Toutes les opérations de maintenance de ce type sont démarrées et terminées dans le délai sélectionné. Si vous ne sélectionnez aucune période, Neptune attribue une période de maintenance de manière arbitraire.
Enable deletion protection (Activer la protection contre la suppression)	(Activée par défaut) La protection contre la suppression empêche la suppression du cluster de bases de données. Vous devez la désactiver explicitement pour pouvoir supprimer le cluster de bases de données.

11. Choisissez Créer une base de données pour lancer le nouveau cluster de bases de données Neptune et son instance principale.

    Dans la console Amazon Neptune, le nouveau cluster de bases de données s'affiche dans la liste des bases de données. Le cluster de base de données a le statut Creating (Création) jusqu'à ce qu'il soit créé et prêt à l'emploi. Lorsque le statut devient Available (Disponible), vous pouvez vous connecter à l'instance principale de votre cluster de base de données. En fonction du stockage et de la classe d'instance de bases de données alloués, la mise à disposition des nouvelles instances de bases de données peut nécessiter plusieurs minutes.

    Pour afficher le cluster que vous venez de créer, choisissez la vue Bases de données dans la console Neptune.

    > **Note**
    >
    > Si vous supprimez toutes les instances de base de données Neptune d'un cluster de base de données à l'aide de AWS Management Console, la console supprime

automatiquement le cluster de base de données lui-même. Si vous utilisez le SDK AWS CLI ou, vous devez supprimer le cluster de base de données manuellement après avoir supprimé sa dernière instance.

Notez la valeur du point de terminaison du cluster. Vous en aurez besoin pour vous connecter à votre cluster de bases de données Neptune.

Arrêt et démarrage d'un cluster de bases de données Amazon Neptune.

L'arrêt et le démarrage des clusters Amazon Neptune vous permettent de maîtriser les coûts liés aux environnements de développement et de test. Vous pouvez arrêter temporairement toutes les instances de base de données de votre cluster au lieu de configurer et de détruire toutes les instances de base de données chaque fois que vous utilisez le cluster.

Rubriques

- Présentation de l'arrêt et du démarrage d'un cluster de bases de données Neptune
- Arrêt d'un cluster de bases de données Neptune
- Démarrage d'un cluster de bases de données Neptune arrêté

Présentation de l'arrêt et du démarrage d'un cluster de bases de données Neptune

Pendant les périodes où vous n'avez pas besoin d'un cluster Neptune, vous pouvez arrêter toutes les instances du cluster en une seule opération. Vous pouvez à tout moment redémarrer le cluster dès que vous avez besoin de l'utiliser. Le démarrage et l'arrêt simplifie les processus de configuration et de destruction des clusters utilisés à des fins de développement, de test ou d'activités similaires qui ne nécessitent pas une disponibilité continue. Vous pouvez y parvenir en AWS Management Console une seule action, quel que soit le nombre d'instances présentes dans le cluster.

Pendant que votre cluster de bases de données est à l'arrêt, vous êtes facturé uniquement pour le stockage du cluster, des instantanés manuels et des sauvegardes automatiques dans le cadre de votre fenêtre de rétention spécifiée. Aucune heure d'instance de base de données ne vous est facturée.

Après sept jours, Neptune redémarre automatiquement votre cluster de bases de données pour éviter qu'il ne passe à côté de mises à jour de maintenance obligatoires.

Afin de limiter les frais pour un cluster Neptune à faible charge, vous pouvez arrêter ce cluster plutôt que de supprimer tous ses réplicas en lecture. Pour les clusters comportant plus d'une ou deux instances, la suppression et la recréation fréquentes des instances de base de données ne sont pratiques qu'à l' AWS CLI aide de l'API Neptune, et les suppressions peuvent également être difficiles à effectuer dans le bon ordre. Par exemple, vous devez supprimer tous les réplicas en lecture avant de supprimer l'instance principale pour éviter d'activer le mécanisme de basculement.

Évitez de démarrer et d'arrêter votre cluster de bases de données s'il doit s'exécuter en permanence, mais que vous souhaitez réduire sa capacité. Si votre cluster est trop coûteux ou sous-utilisé, vous pouvez supprimer une ou plusieurs instances de base de données ou modifier vos instances de base de données afin qu'elles utilisent une classe d'instance plus petite, mais vous ne pouvez pas arrêter une instance de base de données individuelle.

Arrêt d'un cluster de bases de données Neptune

Lorsque vous ne l'utilisez pas pendant un certain temps, vous pouvez arrêter un cluster de bases de données Neptune en cours d'exécution, puis le redémarrer lorsque vous en avez besoin. Pendant que le cluster est à l'arrêt, vous êtes facturé pour le stockage du cluster, des instantanés manuels et des sauvegardes automatiques dans le cadre de votre fenêtre de rétention spécifiée, mais pas pour les heures d'instance de base de données.

L'opération d'arrêt stoppe les instances de réplica en lecture du cluster avant de stopper l'instance principale, pour éviter l'activation du mécanisme de basculement.

Arrêt d'un cluster de base de données à l'aide du AWS Management Console

Pour utiliser le AWS Management Console pour arrêter un cluster Neptune

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le panneau de navigation, sélectionnez Bases de données, choisissez un cluster. Vous pouvez effectuer l'opération d'arrêt soit à partir de cette page, soit en accédant à la page de détails du cluster de bases de données que vous voulez arrêter.
3. Dans Actions, choisissez Arrêter.

Arrêt d'un cluster de base de données à l'aide du AWS CLI

Pour arrêter une instance de base de données à l'aide de AWS CLI, appelez la stop-db-clustercommande en utilisant le `--db-cluster-identifier` paramètre pour identifier le cluster de base de données que vous souhaitez arrêter.

Example

```
aws neptune stop-db-cluster --db-cluster-identifier mydbcluster
```

Arrêt d'un cluster de bases de données à l'aide de l'API de gestion Neptune

Pour arrêter une instance de base de données à l'aide de l'API de gestion Neptune, appelez l'DBClusterAPI Stop et utilisez le `DBClusterIdentifier` paramètre pour identifier le cluster de base de données que vous souhaitez arrêter.

Ce qui peut se produire lorsqu'un cluster de bases de données est arrêté

- Vous pouvez le restaurer à partir d'un instantané (voir Restauration à partir d'un instantané de cluster de base de données).
- Vous ne pouvez pas modifier la configuration du cluster de bases de données ou de ses instances de base de données.
- Vous ne pouvez pas ajouter des instances de base de données au cluster ni supprimer des instances de base de données du cluster.
- Vous ne pouvez pas supprimer le cluster si des instances de base de données lui sont encore associées.
- En général, vous devez redémarrer un cluster de bases de données arrêté pour effectuer la plupart des actions administratives.
- Neptune appliquera les opérations de maintenance planifiée au cluster arrêté dès qu'il est redémarré. N'oubliez pas qu'après sept jours, Neptune redémarre automatiquement un cluster arrêté afin qu'il n'accumule pas trop de retard au niveau de l'état de la maintenance.
- Neptune n'effectue aucune sauvegarde automatisée d'un cluster de bases de données arrêté, car les données sous-jacentes ne peuvent pas changer pendant que le cluster est arrêté.
- Neptune n'étend pas la période de rétention des sauvegarde pour le cluster de bases de données pendant qu'il est arrêté.

Démarrage d'un cluster de bases de données Neptune arrêté

Vous pouvez uniquement démarrer un cluster de bases de données Neptune qui est à l'état arrêté. Lorsque vous démarrez le cluster, toutes ses instances de base de données redeviennent disponibles. Le cluster conserve ses paramètres de configuration, notamment les points de terminaison, les groupes de paramètres et les groupes de sécurité VPC.

Démarrage d'un cluster de base de données arrêté à l'aide du AWS Management Console

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le panneau de navigation, sélectionnez Bases de données, choisissez un cluster. Vous pouvez effectuer l'opération de démarrage soit à partir de cette page, soit en accédant à la page de détails du cluster de bases de données.
3. Dans Actions, choisissez Start (Démarrer).

Démarrage d'un cluster de base de données arrêté à l'aide du AWS CLI

Pour démarrer un cluster de base de données arrêté à l'aide du AWS CLI, appelez la start-db-clustercommande à l'aide du `--db-cluster-identifier` paramètre pour spécifier le cluster de base de données arrêté que vous souhaitez démarrer. Indiquez le nom de cluster que vous avez choisi lors de la création du cluster de bases de données ou utilisez un nom d'instance de base de données que vous avez choisi, en ajoutant `-cluster` à la fin de celui-ci.

Example

```
aws neptune start-db-cluster --db-cluster-identifier mydbcluster
```

Démarrage d'un cluster de bases de données arrêté à l'aide de l'API de gestion Neptune

Pour démarrer un cluster de base de données Neptune à l'aide de l'API de gestion Neptune, appelez l'DBClusterAPI Start à l'aide du `DBCluster` paramètre pour spécifier le cluster de base de données arrêté que vous souhaitez démarrer. Indiquez le nom de cluster que vous avez choisi lors de la création du cluster de bases de données ou utilisez un nom d'instance de base de données que vous avez choisi, en ajoutant `-cluster` à la fin de celui-ci.

Vider un cluster de bases de données Amazon Neptune à l'aide de l'API de réinitialisation rapide

L'API REST de réinitialisation rapide Neptune vous permet de réinitialiser un graphe Neptune rapidement et facilement en supprimant toutes ses données.

Pour ce faire, utilisez la magie linéaire %db_reset dans un bloc-notes Neptune.

- Dans la plupart des cas, une opération de réinitialisation rapide se termine en quelques minutes. La durée peut varier légèrement en fonction de la charge exécutée sur le cluster au lancement de l'opération.
- Une opération de réinitialisation rapide n'entraîne pas d'E/S supplémentaires.
- La taille du volume de stockage ne diminue pas après une réinitialisation rapide. Au lieu de cela, le stockage est réutilisé à mesure que de nouvelles données sont insérées. Par conséquent, les tailles de volume des instantanés créés avant et après une opération de réinitialisation rapide sont les mêmes. Les tailles de volume des clusters restaurés à l'aide des instantanés créés avant et après une opération de réinitialisation rapide sont également les mêmes
- Dans le cadre de l'opération de réinitialisation, toutes les instances du cluster de bases de donnes sont redémarrées.

> **Note**
>
> Dans de rares cas, ces redémarrages de serveur peuvent également entraîner un basculement du cluster.

> **Important**
>
> L'utilisation de la réinitialisation rapide peut interrompre l'intégration de votre cluster de bases de données Neptune avec d'autres services. Par exemple :
>
> - La réinitialisation rapide supprime toutes les données de flux de votre base de données et réinitialise complètement les flux. En d'autres termes, les consommateurs de flux risquent de ne plus fonctionner sans une nouvelle configuration.
> - La réinitialisation rapide supprime toutes les métadonnées relatives aux ressources d' SageMaker IA utilisées par Neptune ML, y compris les tâches et les points de terminaison. Ils existent toujours dans l' SageMaker IA, et vous pouvez continuer à utiliser les points de

terminaison SageMaker IA existants pour les requêtes d'inférence Neptune ML, mais la APIs gestion de Neptune ML ne fonctionne plus avec eux.

- Les intégrations telles que l' full-text-searchintégration avec ElasticSearch sont également annulées par une réinitialisation rapide et doivent être rétablies manuellement avant de pouvoir être réutilisées.

Pour supprimer toutes les données d'un cluster de bases de données Neptune à l'aide de l'API

1. Tout d'abord, générez un jeton que vous pourrez ensuite utiliser pour réinitialiser la base de données. Cette étape vise à empêcher quiconque de réinitialiser accidentellement une base de données.

   Pour ce faire, envoyez une demande `HTTP POST` au point de terminaison `/system` de l'instance d'enregistreur du cluster de bases de données afin de spécifier l'action `initiateDatabaseReset`.

   Voici la commande `curl` qui utilise le type de contenu JSON :

   ```
   curl -X POST \
     -H 'Content-Type: application/json' \
         https://your_writer_instance_endpoint:8182/system \
     -d '{ "action" : "initiateDatabaseReset" }'
   ```

   Ou celle qui utilise le type de contenu `x-www-form-urlencoded` :

   ```
   curl -X POST \
     -H 'Content-Type: application/x-www-form-urlencoded' \
         https://your_writer_instance_endpoint:8182/system \
     -d 'action=initiateDatabaseReset '
   ```

   La demande `initiateDatabaseReset` renvoie le jeton de réinitialisation dans sa réponse JSON, comme suit :

   ```
   {
     "status" : "200 OK",
     "payload" : {
       "token" : "new_token_guid"
     }
   ```

```
}
```

Le jeton reste valide pendant une heure (60 minutes) après son émission.

Si vous envoyez la demande à une instance de lecteur ou au point de terminaison de statut, Neptune génère une exception `ReadOnlyViolationException`.

Si vous envoyez plusieurs demandes `initiateDatabaseReset`, seul le dernier jeton généré est valide pour la deuxième étape, étape au cours de laquelle vous effectuez réellement la réinitialisation.

Si le serveur redémarre juste après la demande `initiateDatabaseReset`, le jeton généré n'est plus valide, et vous devez envoyer une nouvelle demande pour obtenir un nouveau jeton.

2. Ensuite, envoyez une demande `performDatabaseReset` avec le jeton que vous avez obtenu via `initiateDatabaseReset` au point de terminaison `/system` de l'instance d'enregistreur du cluster de bases de données. Cette action supprime toutes les données du cluster de bases de données.

   Voici la commande `curl` qui utilise le type de contenu JSON :

   ```
   curl -X POST \
     -H 'Content-Type: application/json' \
         https://your_writer_instance_endpoint:8182/system \
     -d '{
           "action" : "performDatabaseReset",
           "token" : "token_guid"
         }'
   ```

   Ou celle qui utilise le type de contenu `x-www-form-urlencoded` :

   ```
   curl -X POST \
     -H 'Content-Type: application/x-www-form-urlencoded' \
         https://your_writer_instance_endpoint:8182/system \
     -d 'action=performDatabaseReset&token=token_guid'
   ```

   La demande renvoie une réponse JSON. Si la demande est acceptée, la réponse est la suivante :

   ```
   {
   ```

```
  "status" : "200 OK"
}
```

Si le jeton que vous avez envoyé ne correspond pas à celui qui a été émis, la réponse est la suivante :

```
{
  "code" : "InvalidParameterException",
  "requestId":"token_guid",
  "detailedMessage" : "System command parameter 'token' : 'token_guid' does not
 match database reset token"
}
```

Si la demande est acceptée et que la réinitialisation commence, le serveur redémarre et supprime les données. Vous ne pouvez pas envoyer d'autres demandes au cluster de bases de données pendant sa réinitialisation.

Utilisation de l'API de réinitialisation rapide avec l'authentification IAM

Si l'authentification IAM est activée sur le cluster de bases de données, vous pouvez utiliser awscurl pour envoyer des commandes de réinitialisation rapide authentifiées à l'aide de ce mode d'authentification :

Utilisation d'awscurl pour envoyer des demandes de réinitialisation rapide avec l'authentification IAM

1. Définissez correctement les variables d'environnement `AWS_ACCESS_KEY_ID` et `AWS_SECRET_ACCESS_KEY` (ainsi qu'`AWS_SECURITY_TOKEN` si vous utilisez des informations d'identification temporaires).
2. Une demande `initiateDatabaseReset` se présente comme suit :

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "initiateDatabaseReset" }'
```

3. Une demande `performDatabaseReset` se présente comme suit :

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "performDatabaseReset" }'
```

Utilisation de la magie linéaire `%db_reset` du workbench Neptune pour réinitialiser un cluster de bases de données

Le workbench Neptune intègre une magie linéaire `%db_reset` qui vous permet de réinitialiser rapidement la base de données dans un bloc-notes Neptune.

Si vous invoquez la magie sans aucun paramètre, un écran vous demande si vous souhaitez supprimer toutes les données du cluster. Vous devez également cocher une case afin de confirmer que vous comprenez que les données du cluster ne seront plus disponibles une fois que vous les aurez supprimées. À ce stade, vous pouvez choisir de continuer et de supprimer les données, ou d'annuler l'opération.

Une option plus risquée consiste à invoquer `%db_reset` avec l'option `--yes` ou `-y`, ce qui entraîne la suppression sans invite supplémentaire.

Vous pouvez également effectuer la réinitialisation en deux étapes, tout comme avec l'API REST :

```
%db_reset --generate-token
```

La réponse est :

```
{
  "status" : "200 OK",
  "payload" : {
    "token" : "new_token_guid"
  }
}
```

Continuez avec :

```
%db_reset --token new_token_guid
```

La réponse est :

```
{
  "status" : "200 OK"
}
```

Codes d'erreur courants pour les opérations de réinitialisation rapide

Code d'erreur Neptune	Statut HTTP	Message	Exemple
`InvalidParameterException`	400	La valeur « » du paramètre de commande système *`action`* « » n'est pas prise en charge *`XXX`*	Paramètre non valide
`InvalidParameterException`	400	Trop de valeurs ont été fournies pour : *`action`*	Une demande de réinitialisation rapide avec plusieurs actions envoyée avec l'en-tête « Content-Type:Application/ » x-www-form-urlencoded
`InvalidParameterException`	400	Champ 'action' en double	Demande de réinitialisation rapide avec plusieurs actions, envoyée avec l'en-tête 'Content-Type: application/json'
`MethodNotAllowedException`	400	Mauvais itinéraire :/*`bad_endpoint`*	Demande envoyée à un point de terminaison incorrect
`MissingParameterException`	400	Paramètres obligatoires manquants : [action]	Une demande de réinitialisation rapide ne contient pas le paramètre 'action' obligatoire

Code d'erreur Neptune	Statut HTTP	Message	Exemple
`ReadOnlyViolationException`	400	Les écritures ne sont pas autorisées sur une instance de réplica en lecture	Une demande de réinitialisation rapide a été envoyée à un point de terminaison de lecteur ou de statut
`AccessDeniedException`	403	Jeton d'authentification manquant	Une demande de réinitialisation rapide a été envoyée sans les signatures correctes à un point de terminaison de base de données sur lequel l'authentification IAM est activée
`ServerShutdownException`	500	La réinitialisation de la base de données est en cours. Réessayez la requête une fois que le cluster sera disponible.	Lorsque la réinitialisation rapide commence, les requêtes Gremlin/Sparql existantes et entrantes échouent.

Ajout d'instances de lecteur Neptune à un cluster de bases de données

Les clusters de bases de données Neptune comprennent une instance de base de données principale et jusqu'à 15 instances de lecteur Neptune. L'instance de base de données principale prend en charge les opérations de lecture et d'écriture, et effectue toutes les modifications de données dans le volume de cluster. Les instances de lecteur Neptune se connectent au même volume de stockage que l'instance de base de données principale et prennent uniquement en charge les opérations de lecture.

Utilisez des instances de lecteur pour décharger les charges de travail en lecture de l'instance de base de données principale.

Nous vous recommandons de répartir l'instance principale et les lecteurs Neptune de votre cluster de bases de données sur plusieurs zones de disponibilité afin d'améliorer la disponibilité du cluster.

La section suivante explique comment créer une instance de lecteur dans votre cluster de bases de données.

Création d'une instance de lecteur Neptune à l'aide de la console

Après avoir créé l'instance principale pour votre cluster de bases de données Neptune, vous pouvez ajouter des instances de lecteur Neptune supplémentaires à l'aide de la console Neptune.

Pour créer une instance de lecteur Neptune à l'aide du AWS Management Console

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Sélectionnez le cluster de bases de données dans lequel vous souhaitez créer l'instance de lecteur.
4. Choisissez Actions, puis Ajouter un lecteur.
5. Sur la page Créer un réplica d'instance de base de données, spécifiez les options de votre réplica Neptune. Le tableau suivant présente les paramètres d'un réplica en lecture Neptune.

Pour cette option...	Procédez comme suit
Classe d'instances de base de données	Choisissez une classe d'instances de base de données qui définit les exigences de mémoire et de traitement pour le réplica Neptune. Pour obtenir la liste actuelle des classes d'instances de base de données proposées par Neptune dans différentes régions, consultez la page de tarification Neptune.
Zone de disponibilité	Spécifiez une zone de disponibilité. Choisissez une zone différente de celle de l'instance de base de données principale. La liste n'inclut que les zones de disponibilité qui sont mappées par le groupe de sous-réseaux de base de données pour le cluster de base de données.
Chiffrement	Activez ou désactivez le chiffrement.
Source du réplica en lecture	Choisissez l'identifiant de l'instance principale pour laquelle créer un réplica Neptune.

Pour cette option...	Procédez comme suit
Identifiant d'instance de base de données	Saisissez un nom pour l'instance qui est unique pour votre compte dans la région que vous avez sélection née. Vous pouvez choisir de complexifier le nom, par exemple en incluant la zone de disponibilité sélection née, comme `neptune-us-east-1c` .
Port de la base de données	Numéro de port au niveau duquel la base de données accepte les connexions.
Groupe de paramètres de base de données	Groupe de paramètres de cette instance.
Exportations des journaux	Choisissez les journaux que vous souhaitez publier, le cas échéant.
Mise à niveau automatique de versions mineures	Choisissez Oui afin de permettre à votre réplica Neptune de recevoir automatiquement les mises à niveau des versions mineures du moteur de base de données Neptune quand elles seront disponibles. L'option Mise à niveau automatique de versions mineures s'applique uniquement aux mises à niveau mineures. Elle ne s'applique pas aux correctifs de maintenance du moteur, qui sont toujours appliqués automatiquement pour maintenir la stabilité du système.

6. Choisissez Créer un réplica en lecture pour créer l'instance de réplica Neptune.

Pour supprimer une instance de lecteur Neptune d'un cluster de bases de données, suivez les instructions de la section Suppression d'une instance de base de données dans Amazon Neptune.

Modification d'un cluster de bases de données Neptune à l'aide de la console

Lorsque vous modifiez une instance de base de données à l'aide du AWS Management Console, vous pouvez choisir d'appliquer les modifications immédiatement en sélectionnant Appliquer immédiatement. Si vous choisissez d'appliquer les modifications immédiatement, vos nouvelles modifications et les modifications placées dans la file d'attente des modifications en attente sont appliquées immédiatement.

Si vous ne choisissez pas d'appliquer les modifications immédiatement, les modifications sont placées dans la file d'attente des modifications en attente. Au cours de la fenêtre de maintenance suivante, les modifications en attente sont appliquées.

> **Important**
>
> Si les modifications en attente exigent un temps d'arrêt, le fait de choisir de les appliquer immédiatement peut entraîner un temps d'arrêt imprévu pour l'instance de base de données en question. Il n'y a pas de durée d'indisponibilité pour les autres instances de base de données du cluster de bases de données.

> **Note**
>
> Lorsque vous modifiez un cluster de bases de données dans Neptune, le paramètre Appliquer immédiatement concerne uniquement les modifications apportées à l'identifiant de cluster de bases de données et à l'authentification de la base de données IAM. Toutes les autres modifications sont appliquées immédiatement, quelle que soit la valeur du paramètre Appliquer immédiatement.

Pour modifier un cluster de base de données à l'aide de la console

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le volet de navigation, choisissez Clusters, puis sélectionnez le cluster de bases de données que vous souhaitez modifier.

3. Choisissez Actions, puis Modifier le cluster. La page Modify DB cluster (Modifier le cluster DB) s'affiche.
4. Modifiez les paramètres de votre choix.

> **Note**
>
> Sur la console, certaines modifications au niveau de l'instance s'appliquent uniquement à l'instance de base de données active, tandis que d'autres s'appliquent à l'intégralité du cluster de bases de données. Pour modifier un paramètre qui modifie l'intégralité du cluster de bases de données au niveau de l'instance sur la console, suivez les instructions de Modification d'une instance de base de données dans un cluster de base de données.

5. Lorsque toutes les modifications vous conviennent, choisissez Continuer et vérifiez le résumé.
6. Pour immédiatement appliquer les modifications, sélectionnez Apply Immediately (Appliquer immédiatement).
7. Sur la page de confirmation, examinez vos modifications. Si elles sont correctes, choisissez Modifier le cluster pour enregistrer vos modifications.

   Pour éditer vos modifications, cliquez sur Back (Retour), ou pour annuler vos modifications, choisissez Cancel (Annuler).

Modification d'une instance de base de données dans un cluster de base de données

Pour modifier une instance DB dans un cluster DB à l'aide de la console

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le panneau de navigation, choisissez Instances, puis l'instance de base de données que vous souhaitez modifier.
3. Choisissez Actions d'instance, puis Modifier. La page Modifier l'instance de base de données s'affiche.
4. Modifiez les paramètres de votre choix.

> **Note**
>
> Certains paramètres s'appliquent à l'intégralité du cluster de bases de données et doivent être modifiés au niveau du cluster. Pour modifier ces paramètres, suivez les instructions de la section Modification d'un cluster de bases de données Neptune à l'aide de la console.
>
> Dans le AWS Management Console, certaines modifications au niveau de l'instance s'appliquent uniquement à l'instance de base de données actuelle, tandis que d'autres s'appliquent à l'ensemble du cluster de bases de données.

5. Lorsque toutes les modifications vous conviennent, choisissez Continuer et vérifiez le résumé.
6. Pour immédiatement appliquer les modifications, sélectionnez Apply Immediately (Appliquer immédiatement).
7. Sur la page de confirmation, examinez vos modifications. Si elles sont correctes, choisissez Modification d'une instance de base de données pour enregistrer vos modifications.

   Pour éditer vos modifications, cliquez sur Back (Retour), ou pour annuler vos modifications, choisissez Cancel (Annuler).

Performances et mise à l'échelle dans Amazon Neptune

Les clusters de bases de données et les instances de base de données Neptune se mettent à l'échelle à trois niveaux différents :

- Dimensionnement du stockage
- Mise à l'échelle d'une instance ;
- Dimensionnement en lecture

Mise à l'échelle du stockage dans Neptune

Le stockage Neptune procède à une mise à l'échelle automatique avec les données de votre volume de cluster. À mesure que vos données augmentent, le volume de stockage de votre cluster augmente, jusqu'à 128 TiB dans toutes les régions prises en charge, à l'exception de la Chine GovCloud, où il est limité à 64 TiB.

La taille de votre volume de cluster est vérifiée toutes les heures afin de déterminer vos coûts de stockage.

Le stockage consommé par votre base de données Neptune est facturé par tranches de Go par mois et par tranche de consommation par votre base de données Neptune, et vous I/Os consumed are billed in per million request increments. You pay only for the storage and I/Os n'avez pas besoin de le provisionner à l'avance.

Pour plus d'informations, consultez la page produit Neptune.

Mise à l'échelle des instances dans Neptune

Vous pouvez mettre à l'échelle le cluster de bases de données Neptune en modifiant la classe de chaque instance de base de données dans le cluster de bases de données. Neptune prend en charge plusieurs classes d'instances de base de données à des valeurs optimales.

Mise à l'échelle des lectures dans Neptune

Vous pouvez procéder à une mise à l'échelle en lecture de votre cluster de bases de données Neptune en créant jusqu'à 15 réplicas Neptune dans le cluster de bases de données. Chaque réplica Neptune renvoie les mêmes données du volume de cluster avec un retard de réplica minimal (souvent très inférieur à 100 millisecondes, après que l'instance principale a écrit une mise à jour).

Tandis que votre trafic en lecture augmente, vous pouvez créer des réplicas Neptune additionnels et vous y connecter directement pour répartir la charge de lecture de votre cluster de bases de données. Les réplicas Neptune n'ont pas à être de la même classe d'instances de base de données que l'instance principale.

Pour en savoir plus sur l'ajout de réplicas Neptune à un cluster de bases de données, consultez Ajout d'instances de lecteur.

Autoscaling du nombre de réplicas dans un cluster de bases de données Amazon Neptune

Vous pouvez utiliser l'autoscaling Neptune pour ajuster automatiquement le nombre de réplicas Neptune dans un cluster de bases de données afin de répondre à vos exigences en matière de connectivité et de charge de travail. L'autoscaling permet au cluster de bases de données Neptune de gérer l'augmentation de la charge de travail, puis, lorsque la charge de travail diminue, il supprime les réplicas inutiles afin que vous ne payiez pas la capacité inutilisée.

Vous ne pouvez utiliser l'auto-scaling qu'avec un cluster de bases de données Neptune qui possède déjà une instance d'enregistreur principale et au moins une instance de réplica en lecture (voir Clusters de bases de données et instances de base de données Amazon Neptune). En outre, toutes les instances de réplica en lecture du cluster doivent être dans un état disponible. Si un réplica en lecture présente un état autre que disponible, l'autoscaling Neptune ne fait rien tant que tous les réplicas en lecture du cluster ne sont pas disponibles.

Consultez Créer un cluster Neptune si vous devez créer un cluster.

À l'aide du AWS CLI, vous définissez et appliquez une politique de dimensionnement au cluster de bases de données. Vous pouvez également utiliser le AWS CLI pour modifier ou supprimer votre politique d'auto-scaling. Cette politique spécifie les paramètres d'autoscaling suivants :

- Nombre minimal et maximal de réplicas dans le cluster.
- `ScaleOutCooldown`Intervalle entre l'activité de dimensionnement par ajout de répliques et `ScaleInCooldown` intervalle entre l'activité de dimensionnement par suppression de répliques.
- La CloudWatch métrique et la valeur de déclenchement de la métrique pour la mise à l'échelle à la hausse ou à la baisse.

La fréquence des actions d'autoscaling Neptune est réduite de plusieurs manières :

- Au départ, pour que l'autoscaling puisse ajouter ou supprimer un lecteur, le seuil élevé de l'alarme `CPUUtilization` doit être activé pendant au moins trois minutes ou le seuil bas de l'alarme doit être activé pendant au moins 15 minutes.
- Après ce premier ajout ou cette première suppression, la fréquence des actions d'autoscaling Neptune suivantes est limitée par les paramètres `ScaleOutCooldown` et `ScaleInCooldown` de la politique d'autoscaling.

Si la CloudWatch métrique que vous utilisez atteint le seuil élevé que vous avez spécifié dans votre politique, si l'`ScaleOutCooldown`intervalle s'est écoulé depuis la dernière action d'auto-scaling, et si votre cluster de bases de données ne possède pas encore le nombre maximum de répliques que vous avez défini, Neptune auto-scaling crée un nouveau réplica en utilisant le même type d'instance que l'instance principale du cluster de base de données.

De même, si la métrique atteint le seuil bas que vous avez spécifié, si l'intervalle `ScaleInCooldown` s'est écoulé depuis la dernière action d'auto-scaling et si votre cluster de bases de données possède un nombre de réplicas supérieur au nombre minimum que vous avez spécifié, l'auto-scaling Neptune supprime l'un des réplicas.

> **Note**
>
> L'autoscaling Neptune supprime uniquement les réplicas qu'il a créés. Les réplicas préexistants ne sont pas supprimés.

À l'aide du paramètre de cluster de bases de données neptune_autoscaling_config, vous pouvez également spécifier le type d'instance des réplicas en lecture créés par l'auto-scaling Neptune, les fenêtres de maintenance de ces réplicas et les balises à associer à chacun d'eux. Vous fournissez ces paramètres de configuration dans une chaîne JSON en tant que valeur du paramètre `neptune_autoscaling_config`, comme suit :

```
"{
  \"tags\": [
    { \"key\" : \"reader tag-0 key\", \"value\" : \"reader tag-0 value\" },
    { \"key\" : \"reader tag-1 key\", \"value\" : \"reader tag-1 value\" },
  ],
  \"maintenanceWindow\" : \"wed:12:03-wed:12:33\",
  \"dbInstanceClass\" : \"db.r5.xlarge\"
}"
```

Notez que les guillemets de la chaîne JSON doivent tous être précédés d'une barre oblique inverse (\). Tous les espaces blancs dans la chaîne sont facultatifs, comme d'habitude.

Les trois paramètres de configuration non spécifiés dans le paramètre `neptune_autoscaling_config` sont copiés à partir de la configuration de l'instance d'enregistreur principale du cluster de bases de données.

Lorsque l'autoscaling ajoute une nouvelle instance de réplica en lecture, il insère le préfixe `autoscaled-reader` dans l'ID de l'instance de base de données (par exemple, `autoscaled-reader-7r7t7z3lbd-20210828`). Il ajoute également une balise à chaque réplica en lecture qu'il crée avec la clé `autoscaled-reader` et la valeur `TRUE`. Vous pouvez voir cette balise dans l'onglet Balises de la page de détails de l'instance de base de données dans la AWS Management Console.

```
"key" : "autoscaled-reader",  "value" : "TRUE"
```

Le niveau de promotion de toutes les instances de réplica en lecture créées par l'auto-scaling est le niveau de priorité le plus bas, qui est 15 par défaut. Cela signifie que pendant un basculement, un réplica ayant une priorité supérieure (par exemple, un réplica créé manuellement) serait promu en premier. Consultez Tolérance aux pannes pour un cluster de bases de données Neptune.

L'auto-scaling de Neptune est implémenté à l'aide d'Application Auto Scaling avec une politique de dimensionnement du suivi des cibles qui utilise une métrique Neptune comme métrique prédéfinie `CPUUtilization` CloudWatch .

Utilisation de l'auto-scaling dans un cluster de bases de données Neptune sans serveur

Neptune sans serveur répond beaucoup plus rapidement que l'autoscaling Neptune lorsque la demande dépasse la capacité d'une instance, et augmente la capacité de l'instance au lieu d'en ajouter une autre. Là où l'autoscaling est conçu pour répondre à des augmentations ou à des baisses relativement stables de la charge de travail, le système sans serveur excelle dans la gestion des pics et des fluctuations rapides de la demande.

En comprenant leurs points forts, vous pouvez combiner l'autoscaling et le mode sans serveur pour créer une infrastructure flexible capable de gérer efficacement l'évolution de votre charge de travail et de répondre à la demande tout en minimisant les coûts.

Pour permettre à l'autoscaling de fonctionner efficacement avec le mode sans serveur, il est important de configurer le paramètre `maxNCU` de votre cluster sans serveur pour qu'il soit suffisamment élevé afin de faire face aux pics et aux brèves variations de la demande. Dans le cas contraire, les modifications transitoires ne déclencheront pas la mise à l'échelle sans serveur, ce qui pourra entraîner l'activation de nombreuses instances supplémentaires inutiles. Si la valeur `maxNCU` est suffisamment élevée, la mise à l'échelle sans serveur pourra gérer ces modifications plus rapidement et à moindre coût.

Comment activer l'autoscaling pour Amazon Neptune

Le dimensionnement automatique ne peut être activé que pour un cluster de bases de données Neptune à l'aide de l' AWS CLI. Vous ne pouvez pas activer l'autoscaling à l'aide de la AWS Management Console.

De plus, l'autoscaling n'est pas pris en charge dans les régions Amazon suivantes :

- Afrique (Le Cap) : `af-south-1`
- Moyen-Orient (EAU) : `me-central-1`
- AWS GovCloud (USA Est) : `us-gov-east-1`
- AWS GovCloud (USA Ouest) : `us-gov-west-1`

L'activation de l'autoscaling pour un cluster de bases de données Neptune implique trois étapes :

1. Enregistrement de votre cluster de bases de données avec Application Auto Scaling

La première étape pour activer l'auto-scaling pour un cluster de base de données Neptune consiste à enregistrer le cluster auprès d'Application Auto Scaling, en utilisant le AWS CLI ou l'un des outils Application Auto Scaling. SDKs Le cluster doit déjà avoir une instance principale et au moins une instance de réplica en lecture :

Par exemple, pour enregistrer un cluster à dimensionner automatiquement avec une à huit répliques supplémentaires, vous pouvez utiliser la AWS CLI `register-scalable-target`commande suivante :

```
aws application-autoscaling register-scalable-target \
  --service-namespace neptune \
  --resource-id cluster:(your DB cluster name) \
  --scalable-dimension neptune:cluster:ReadReplicaCount \
  --min-capacity 1 \
  --max-capacity 8
```

Cela est équivalent à l'utilisation de l'opération `RegisterScalableTarget` de l'API Application Auto Scaling.

La commande AWS CLI `register-scalable-target` utilise les paramètres suivants :

- **`service-namespace`** : défini sur `neptune`.

Ce paramètre est équivalent au paramètre `ServiceNamespace` de l'API Application Auto Scaling.

- **`resource-id`** : définissez ce paramètre sur l'identifiant de ressource de votre cluster de bases de données Neptune. Le type de ressource est `cluster`, suivi du signe deux-points (`:`), puis du nom de votre cluster de bases de données.

  Ce paramètre est équivalent au paramètre `ResourceID` de l'API Application Auto Scaling.

- **`scalable-dimension`** : la dimension pouvant être mise à l'échelle dans ce cas est le nombre d'instances de réplica dans le cluster de bases de données. Vous devez donc définir ce paramètre sur `neptune:cluster:ReadReplicaCount`.

  Ce paramètre est équivalent au paramètre `ScalableDimension` de l'API Application Auto Scaling.

- **`min-capacity`** : nombre minimal d'instances de réplica de base de données de lecteur devant être gérées par Application Auto Scaling. Cette valeur doit être comprise entre 0 et 15 et doit être inférieure ou égale à la valeur spécifiée pour le nombre maximal de réplicas Neptune indiqué dans `max-capacity`. Il doit y avoir au moins un lecteur dans le cluster de bases de données pour que l'autoscaling fonctionne.

  Ce paramètre est équivalent au paramètre `MinCapacity` de l'API Application Auto Scaling.

- **`max-capacity`** : nombre maximal d'instances de réplica de base de données de lecteur dans le cluster de bases de données, y compris les instances préexistantes et les nouvelles instances gérées par Application Auto Scaling. Cette valeur doit être comprise entre 0 et 15 et doit être supérieure ou égale à la valeur spécifiée pour le nombre minimal de réplicas Neptune indiqué dans `min-capacity`.

  Le `max-capacity` AWS CLI paramètre est équivalent au `MaxCapacity` paramètre de l'API Application Auto Scaling.

Lorsque vous enregistrez votre cluster de bases de données, Application Auto Scaling crée un rôle `AWSServiceRoleForApplicationAutoScaling_NeptuneCluster` lié à un service. Pour plus d'informations, consultez Rôles liés aux services pour Application Auto Scaling dans le Guide de l'utilisateur Application Auto Scaling.

2. Définition d'une politique d'autoscaling à utiliser avec votre cluster de bases de données

Une politique de mise à l'échelle de suivi de la cible est définie comme un objet texte JSON qui peut également être enregistré dans un fichier texte. Pour Neptune, cette politique ne peut actuellement utiliser la métrique Neptune qu'en tant que `CPUUtilization` CloudWatch métrique prédéfinie nommée. `NeptuneReaderAverageCPUUtilization`

L'exemple suivant illustre une configuration avec suivi de cible d'une politique de mise à l'échelle pour Neptune :

```
{
  "PredefinedMetricSpecification": { "PredefinedMetricType":
 "NeptuneReaderAverageCPUUtilization" },
  "TargetValue": 60.0,
  "ScaleOutCooldown" : 600,
  "ScaleInCooldown" : 600
}
```

L'élément **`TargetValue`** contient le pourcentage d'utilisation de CPU au-dessus duquel l'autoscaling monte en puissance (en ajoutant des réplicas) et en dessous duquel il procède à une mise à l'échelle horizontale (en supprimant des réplicas). Dans ce cas, le pourcentage cible qui déclenche la mise à l'échelle est `60.0` %.

L'élément **`ScaleInCooldown`** spécifie la durée, en secondes, devant s'écouler entre la fin d'une activité de mise à l'échelle horizontale et le début d'une autre. La durée par défaut est 300 secondes. Ici, la valeur 600 indique qu'au moins dix minutes doivent s'écouler entre la fin de la suppression d'un réplica et le début d'un autre.

L'élément **`ScaleOutCooldown`** spécifie la durée, en secondes, devant s'écouler entre la fin d'une activité de mise à l'échelle verticale et le début d'une autre. La durée par défaut est 300 secondes. Ici, la valeur 600 indique qu'au moins dix minutes doivent s'écouler entre la fin de l'ajout d'un réplica et le début d'un autre.

L'élément **`DisableScaleIn`** est un booléen qui, s'il est présent et défini sur `true`, dèsactive complètement la mise à l'échelle horizontale, ce qui signifie que l'autoscaling peut ajouter des réplicas, mais n'en supprimera jamais aucun. Par défaut, la mise à l'échelle horizontale est activée, et `DisableScaleIn` est défini sur `false`.

Après avoir enregistré votre cluster de bases de données Neptune auprès d'Application Auto Scaling et après avoir défini une politique de mise à l'échelle, appliquez cette dernière au cluster de bases de données Neptune enregistré. Pour ce faire, vous pouvez utiliser la AWS CLI `put-scaling-policy`commande avec des paramètres tels que les suivants :

```
aws application-autoscaling put-scaling-policy \
  --policy-name (name of the scaling policy) \
  --policy-type TargetTrackingScaling \
  --resource-id cluster:(name of your Neptune DB cluster) \
  --service-namespace neptune \
  --scalable-dimension neptune:cluster:ReadReplicaCount \
  --target-tracking-scaling-policy-configuration file://(path to the JSON configuration
 file)
```

Lorsque vous avez appliqué la politique d'autoscaling, l'autoscaling est activé sur votre cluster de bases de données.

Vous pouvez également utiliser la AWS CLI `put-scaling-policy`commande pour mettre à jour une politique d'auto-scaling existante.

Voir également PutScalingPolicyla référence de l'API Application Auto Scaling.

Suppression de l'autoscaling d'un cluster de bases de données Neptune

Pour supprimer l'auto-scaling d'un cluster de base de données Neptune, utilisez les commandes et. AWS CLI delete-scaling-policyderegister-scalable-target

Maintenance du cluster de bases de données Amazon Neptune

Neptune assure la maintenance périodique de toutes les ressources qu'elle utilise, notamment :

- Remplacement du matériel sous-jacent si nécessaire. Cela se produit en arrière-plan, sans que vous ayez à agir, et n'a généralement aucune incidence sur vos opérations.
- Mise à jour du système d'exploitation sous-jacent. Les mises à niveau du système d'exploitation des instances de votre cluster de bases de données visent à améliorer les performances et la sécurité. Vous devez donc généralement les terminer dès que possible. En général, les mises à jour prennent environ 10 minutes. Les mises à jour du système d'exploitation ne modifient pas la version du moteur de base de données ou la classe d'instance de base de données d'une instance de base de données.

  Il est généralement préférable de mettre à jour d'abord les instances de lecteur dans un cluster de bases de données, puis l'instance de scripteur. La mise à jour simultanée des lecteurs et de l'écriture peut provoquer un temps d'arrêt en cas de basculement. Notez que les instances de base de données ne sont pas automatiquement sauvegardées avant une mise à jour du système d'exploitation. Veillez donc à effectuer des sauvegardes manuelles avant d'appliquer une mise à jour du système d'exploitation.
- Mise à jour du moteur de base de données Neptune. Neptune publie régulièrement diverses mises à jour du moteur afin d'introduire de nouvelles fonctionnalités et améliorations et de corriger des bogues.

Numéros de version du moteur

Numérotation des versions avant la version du moteur 1.3.0.0

Avant novembre 2019, Neptune ne prenait en charge qu'une seule version du moteur à la fois, et les numéros de version du moteur prenaient tous la forme `1.0.1.0.200`*`<xxx>`*, où `xxx` était le numéro de correctif. Toutes les nouvelles versions du moteur ont toutes été publiées sous la forme de correctifs pour les versions antérieures.

Depuis novembre 2019, Neptune prend en charge plusieurs versions, ce qui permet aux clients de mieux contrôler leurs chemins de mise à niveau. Par conséquent, la numérotation des versions du moteur a changé.

De novembre 2019 jusqu'à la version 1.3.0.0 du moteur, les numéros de version du moteur comportent 5 parties. Prenez le numéro de version `1.0.2.0.R2` par exemple :

- La première partie était toujours 1.
- La deuxième partie, `0` dans `1.0.2.0.R2`, était le numéro de version principal de la base de données.
- Les troisième et quatrième parties, `1.0.2.0.R2` dans `2.0`, étaient toutes deux des numéros de version mineures.
- La cinquième partie (`R2` dans `1.0.2.0.R2`) était le numéro de correctif.

La plupart des mises à jour étaient des correctifs, et la distinction entre les correctifs et les mises à jour de versions mineures n'était pas toujours claire.

Numérotation des versions à partir de la version 1.3.0.0 du moteur

À partir de la version 1.3.0.0 du moteur, Neptune a changé la façon dont les mises à jour du moteur sont numérotées et gérées.

Les numéros de version du moteur comportent désormais quatre parties, chacune correspondant à un type de version, comme suit :

product-version.*major-version*.*minor-version*.*patch-version*

Non-breaking les modifications du type précédemment publiées sous forme de correctifs sont désormais publiées sous forme de versions mineures que vous pouvez gérer à l'aide du paramètre d'`AutoMinorVersionUpgrade`instance.

Cela signifie que si vous le souhaitez, vous pouvez recevoir une notification chaque fois qu'une nouvelle version mineure est publiée, en vous inscrivant à l'événement `RDS-EVENT-0156` (voir Abonnement à la notification d'événement Neptune).

Les versions de correctifs sont désormais réservées aux corrections ciblées urgentes et sont numérotées à l'aide de la dernière partie du numéro de version (`*.*.*.1`, `*.*.*.2`, etc.).

Différents types de version de moteurs dans Amazon Neptune

Les quatre types de version du moteur correspondant aux quatre parties d'un numéro de version du moteur sont les suivants :

- Version du produit : cela ne change que si le produit subit des modifications profondes et fondamentales en termes de fonctionnalité ou d'interface. La version actuelle du produit Neptune est 1.

- Version majeure : les versions majeures introduisent de nouvelles fonctionnalités importantes et des modifications majeures, et ont généralement une durée de vie utile d'au moins deux ans.
- Version mineure : les versions mineures peuvent contenir de nouvelles fonctionnalités, des améliorations et des corrections de bogues, mais ne contiennent aucune modification majeure. Vous pouvez choisir de les faire appliquer automatiquement ou non lors de la prochaine fenêtre de maintenance, et vous pouvez également choisir d'être averti chaque fois qu'elles sont publiées.
- Version de correctif : les versions de correctif sont publiées uniquement pour corriger des bogues urgents ou apporter des mises à jour de sécurité critiques. Elles contiennent rarement des modifications majeures, et elles sont automatiquement appliquées lors de la fenêtre de maintenance suivante après leur publication.

Mises à jour des versions majeures d'Amazon Neptune

Une mise à jour de version majeure introduit généralement une ou plusieurs nouvelles fonctionnalités importantes et contient souvent des modifications majeures. Sa durée de vie de support est généralement d'environ deux ans. Les versions majeures de Neptune sont répertoriées dans les versions du moteur, avec leur date de sortie et leur date de fin de vie estimée.

Les mises à jour des versions majeures sont entièrement facultatives jusqu'à ce que la version majeure que vous utilisez atteigne la fin de son cycle de vie. Si vous choisissez de passer à une nouvelle version majeure, vous devez installer la nouvelle version vous-même à l'aide de la console Neptune AWS CLI ou de la console Neptune, comme décrit dans. Mises à niveau de version majeure.

Toutefois, si la version majeure que vous utilisez arrive à expiration, vous serez averti que vous devez effectuer une mise à niveau vers une version majeure plus récente. Ensuite, si vous n'effectuez pas de mise à niveau dans le délai de grâce suivant la notification, une mise à niveau vers la version majeure la plus récente est automatiquement planifiée lors de la prochaine fenêtre de maintenance. Pour plus d'informations, consultez Durée de vie des versions du moteur.

Mises à jour des versions mineures d'Amazon Neptune

La plupart des mises à jour du moteur Neptune sont des mises à jour mineures. Elles se produisent assez fréquemment et ne contiennent pas de modifications majeures.

Si le champ `AutoMinorVersionUpgrade` est défini sur `true` dans l'instance de dispositif d'écriture (principale) de votre cluster de base de données, les mises à jour de version mineures sont

appliquées automatiquement à toutes les instances de votre cluster de bases de données lors de la fenêtre de maintenance suivante après leur publication.

Si le champ `AutoMinorVersionUpgrade` est défini sur `false` dans l'instance de dispositif d'écriture de votre cluster de bases de données, ils ne sont appliqués que si vous les installez explicitement.

> **Note**
>
> Les mises à jour des versions mineures sont autonomes (elles ne dépendent pas des mises à jour précédentes de la même version majeure) et cumulatives (elles contiennent toutes les fonctionnalités et tous les correctifs introduits dans les mises à jour des versions mineures précédentes). Cela signifie que vous pouvez installer n'importe quelle mise à jour de version mineure, que vous ayez installé les versions précédentes ou non.

Vous pouvez facilement suivre les sorties de versions mineures en vous abonnant à l'événement `RDS-EVENT-0156` (consultez Abonnement à la notification d'événement Neptune). Vous serez alors averti chaque fois qu'une nouvelle version mineure sera publiée.

De plus, que vous soyez abonné ou non aux notifications, vous pouvez toujours vérifier quelles mises à jour sont en attente.

Mises à jour de version de correctif d'Amazon Neptune

En cas de problèmes de sécurité ou d'autres défauts graves affectant la fiabilité de l'instance, Neptune déploie les correctifs obligatoires. Ils sont appliqués à toutes les instances de votre cluster de bases de données lors de votre prochain créneau de maintenance sans aucune intervention de votre part.

Une version de correctif n'est déployée que lorsque les risques liés à son non-déploiement l'emportent sur les risques et les interruptions de service associés à son déploiement. Les versions de correctif ont lieu peu fréquemment (en général une fois à quelques mois d'intervalle) et nécessitent rarement plus d'une fraction de votre fenêtre de maintenance.

Planification de la durée de vie des versions majeures du moteur Amazon Neptune

Les versions du moteur Neptune atteignent presque toujours leur fin de vie à la fin d'un trimestre du calendrier civil, à l'exception des cas où des problèmes importants de sécurité ou de disponibilité surviennent.

Lorsqu'une version du moteur arrive en fin de vie, vous devez mettre à niveau votre base de données Neptune vers une version plus récente.

En général, les versions du moteur Neptune restent disponibles comme suit :

- Versions mineures du moteur : les versions mineures du moteur restent disponibles pendant au moins six mois après leur sortie.
- Versions majeures du moteur : les versions majeures du moteur restent disponibles pendant au moins 12 mois après leur sortie.

Au moins 3 mois avant la fin de vie d'une version du moteur, AWS vous enverrez une notification automatique par e-mail à l'adresse e-mail associée à votre AWS compte et publierez le même message sur votre AWS Health Dashboard. Cela vous laisse ainsi le temps de planifier et de préparer la mise à niveau.

Lorsqu'une version du moteur arrive en fin de vie, vous ne pouvez plus créer de clusters ni d'instances à l'aide de cette version. L'autoscaling ne peut plus créer d'instances à l'aide de cette version non plus.

Une version du moteur qui arrive à sa date de fin de vie effective est automatiquement mise à niveau pendant une fenêtre de maintenance. Le message qui vous est envoyé trois mois avant la fin de vie de la version du moteur contient des informations sur les implications de cette mise à jour automatique, notamment sur la version de la mise à niveau qui aura lieu automatiquement, l'impact sur vos clusters de bases de données et les actions que nous recommandons.

> **Important**
>
> Vous êtes responsable de la mise à jour constante des versions de votre moteur de base de données. AWS invite tous les clients à mettre à niveau leurs bases de données vers la dernière version du moteur afin de bénéficier des garanties de sécurité, de confidentialité et de disponibilité les plus récentes. Si vous utilisez votre base de données sur un moteur ou

un logiciel non pris en charge après la date d'obsolescence (ce que l'on considère comme un « ancien moteur »), vous êtes exposé à un risque accru de problèmes de sécurité, de confidentialité et d'exploitation, y compris des interruptions de service.
L'exploitation de votre base de données sur n'importe quel moteur est soumise à l'accord régissant votre utilisation des AWS services. Les anciens moteurs ne sont généralement pas disponibles. AWS ne fournit plus de support à l'Legacy Engine et AWS peut imposer des limites à l'accès ou à l'utilisation de tout Legacy Engine à tout moment, s'il est AWS déterminé que l'Legacy Engine présente un risque de sécurité ou de responsabilité, ou un risque de préjudice AWS, pour les Services, ses filiales ou tout tiers. Votre décision de continuer à exécuter votre contenu dans un ancien moteur pourrait rendre votre contenu indisponible, corrompu ou irrécupérable. Les bases de données exécutées sur un ancien moteur sont soumises à des exceptions au contrat de niveau de service (SLA).
LES BASES DE DONNÉES ET LES LOGICIELS ASSOCIÉS EXÉCUTÉS SUR UN ANCIEN MOTEUR CONTIENNENT DES BOGUES, DES ERREURS, DES DÉFAUTS ET DES COMPOSANTS AND/OR NUISIBLES. EN CONSÉQUENCE, ET NONOBSTANT TOUTE DISPOSITION CONTRAIRE DANS LE CONTRAT OU LES CONDITIONS DE SERVICE, AWS FOURNIT L'ANCIEN MOTEUR « TEL QUEL ».

Gestion des mises à jour du moteur de votre cluster de bases de données Neptune

Note

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données. Dans de rares cas, un Multi-AZ basculement peut être nécessaire pour terminer une mise à jour de maintenance sur une instance.
Pour les mises à niveau des versions majeures dont l'application peut prendre plus de temps, vous pouvez utiliser une stratégie de déploiement bleu/vert afin de minimiser les temps d'arrêt.

Détermination de la version de moteur que vous utilisez actuellement

Vous pouvez utiliser la AWS CLI `get-engine-status`commande pour vérifier la version du moteur que votre cluster de base de données utilise actuellement :

```
aws neptunedata get-engine-status
```

La sortie JSON inclut un champ `"dbEngineVersion"` comme celui-ci :

```
  "dbEngineVersion": "1.3.0.0",
```

Vérification des mises à jour en attente et disponibles

Vous pouvez rechercher des mises à jour en attente pour votre cluster de bases de données à l'aide de la console Neptune. Sélectionnez Bases de données dans la colonne de gauche, puis sélectionnez votre cluster de bases de données dans le volet des bases de données. Les mises à jour en attente sont répertoriées dans la colonne Maintenance. Si vous sélectionnez Actions puis Maintenance, trois options s'offrent à vous quant à la marche à suivre :

- Mettre à niveau maintenant.
- Mettre à niveau lors de la fenêtre suivante.
- Différer la mise à niveau.

Vous pouvez répertorier les mises à jour du moteur en attente à l'aide de AWS CLI ce qui suit :

```
aws neptune describe-pending-maintenance-actions \
  --resource-identifier (ARN of your DB cluster)
  --region (your region) \
  --engine neptune
```

Vous pouvez également répertorier les mises à jour du moteur disponibles à l'aide AWS CLI des méthodes suivantes :

```
aws neptune describe-db-engine-versions \
  --region (your region) \
  --engine neptune
```

La liste des mises à jour du moteur disponibles inclut uniquement ces mises à jour ayant un numéro de version supérieur au numéro actuel et pour lequel un chemin de mise à niveau est défini.

Toujours effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code même sans modification majeure.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres modifications majeures.

La meilleure façon de tester une nouvelle version avant de mettre à niveau votre cluster de base de données de production est d'utiliser la solution de Blue-Green déploiement Neptune. Ainsi, vous pouvez exécuter des applications et des requêtes sur la nouvelle version sans affecter votre cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

Fenêtre de maintenance Neptune

La fenêtre de maintenance hebdomadaire est une période de 30 minutes au cours de laquelle les mises à jour planifiées du moteur et les autres modifications du système sont appliquées. La plupart des événements de maintenance se terminent également au cours de la fenêtre de maintenance de 30 minutes, mais des événements de maintenance plus importants peuvent prendre plus de 30 minutes.

Chaque cluster de bases de données dispose d'un créneau de maintenance hebdomadaire de 30 minutes. Si vous ne spécifiez pas d'heure préférée lors de la création du cluster de bases de données, Neptune choisit un jour de semaine au hasard, puis attribue au hasard un créneau de 30 minutes sur un bloc horaire de 8 heures qui varie en fonction de la région.

Voici, par exemple, les blocs horaires de 8 heures pour les fenêtres de maintenance utilisées dans plusieurs régions AWS :

Région	Bloc chronologique
Région USA Ouest (Oregon)	06:00–14:00 UTC
Région US West (N. California)	06:00–14:00 UTC
Région USA Est (Ohio)	03:00–11:00 UTC
Région Europe (Irlande)	22:00–06:00 UTC

La fenêtre de maintenance détermine le moment où les opérations en attente commencent, et la plupart des opérations de maintenance se terminent dans cette fenêtre, mais les tâches de maintenance plus importantes peuvent se poursuivre au-delà de l'heure de fin de la fenêtre.

Modification de la fenêtre de maintenance de cluster de bases de données

Idéalement, votre fenêtre de maintenance devrait tomber au moment où votre cluster est le moins utilisé. Si ce n'est pas le cas pour votre fenêtre actuelle, vous pouvez le déplacer vers un meilleur moment, comme ceci :

Pour modifier la fenêtre de maintenance de votre cluster de bases de données

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le volet de navigation, sélectionnez les bases de données.
3. Choisissez le cluster de bases de données pour lequel vous souhaitez modifier la fenêtre de maintenance.
4. Sélectionnez Modify.

5. Choisissez Afficher plus en bas de la page Modifier le cluster.
6. Dans la section Fenêtre de maintenance préférée, définissez le jour, l'heure et la durée de la fenêtre de maintenance comme vous le souhaitez.
7. Choisissez Suivant.

   Sur la page de confirmation, examinez vos modifications.
8. Pour appliquer immédiatement les modifications à la fenêtre de maintenance, sélectionnez Appliquer immédiatement.
9. Choisissez Soumettre pour enregistrer vos modifications.

   Pour modifier vos modifications, cliquez sur Retour, ou pour annuler vos modifications, choisissez Annuler.

Utilisation AutoMinorVersionUpgrade pour contrôler les mises à jour automatiques des versions mineures

> **Important**
>
> `AutoMinorVersionUpgrade` n'est efficace que pour les mises à niveau de versions mineures supérieures à la version 1.3.0.0 du moteur.
> Les mises à niveau automatiques des versions mineures ne s'appliquent pas aux clusters Neptune qui font partie d'une base de données globale. Vous pouvez définir cette option sur une instance de base de données dans un cluster de base de données global, mais cela n'a aucun effet.

Si le champ `AutoMinorVersionUpgrade` est défini sur `true` dans l'instance de dispositif d'écriture (principale) de votre cluster de base de données, les mises à jour de version mineures sont appliquées automatiquement à toutes les instances de votre cluster de bases de données lors de la fenêtre de maintenance suivante après leur publication.

Si le champ `AutoMinorVersionUpgrade` est défini sur `false` dans l'instance de dispositif d'écriture de votre cluster de bases de données, ils ne sont appliqués que si vous les installez explicitement.

> **Note**
>
> Les versions de correctifs (`*.*.*.1`, `*.*.*.2`, etc.) sont toujours installées automatiquement lors de votre prochaine fenêtre de maintenance, quel que soit le paramètre `AutoMinorVersionUpgrade` défini.

Vous pouvez effectuer le réglage `AutoMinorVersionUpgrade` en utilisant AWS Management Console les options suivantes :

Pour définir **`AutoMinorVersionUpgrade`** à l'aide de la console Neptune

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Sélectionnez l'instance principale (dispositif d'écriture) du cluster de bases de données pour lequel vous souhaitez procéder à la définition de `AutoMinorVersionUpgrade`.
4. Sélectionnez Modifier.
5. Choisissez Afficher plus en bas de la page Modifier le cluster.
6. Au bas de la page développée, choisissez Activer la mise à niveau automatique de la version mineure ou Désactiver la mise à niveau automatique de la version mineure.
7. Choisissez Suivant.

   Sur la page de confirmation, examinez vos modifications.
8. Pour appliquer les modifications à la mise à niveau automatique de version mineure, sélectionnez Appliquer immédiatement.
9. Choisissez Soumettre pour enregistrer vos modifications.

   Pour modifier vos modifications, cliquez sur Retour, ou pour annuler vos modifications, choisissez Annuler.

Vous pouvez également utiliser le AWS CLI pour définir le `AutoMinorVersionUpgrade` champ. Par exemple, pour le définir sur `true`, vous pouvez utiliser une commande comme celle-ci :

```
aws neptune modify-db-instance \
  --db-instance-identifier (the ID of your cluster's writer instance) \
```

```
  --auto-minor-version-upgrade \
  --apply-immediately
```

De même, pour le définir sur `false`, utilisez une commande comme celle-ci :

```
aws neptune modify-db-instance \
  --db-instance-identifier (the ID of your cluster's writer instance) \
  --no-auto-minor-version-upgrade \
  --apply-immediately
```

Installation manuelle des mises à jour de votre moteur Neptune

Installation d'une mise à niveau de version majeure du moteur

Les versions majeures du moteur doivent être installées manuellement. Pour minimiser les temps d'arrêt et laisser suffisamment de temps aux tests et à la validation, le meilleur moyen d'installer une nouvelle version majeure est généralement d'utiliser la solution de Blue-Green déploiement Neptune.

Dans certains cas, vous pouvez également utiliser le CloudFormation modèle avec lequel vous avez créé votre cluster de base de données pour installer une mise à niveau de version majeure (voirUtilisation d'un CloudFormation modèle pour mettre à jour la version du moteur de votre cluster de base de données Neptune).

Si vous souhaitez installer une mise à jour de version majeure immédiatement, vous pouvez utiliser une commande CLI comme celle-ci :

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (identifier for your neptune cluster) \
  --engine neptune \
  --engine-version (the new engine version) \
  --apply-immediately
```

Assurez-vous de spécifier la version du moteur vers laquelle vous voulez effectuer la mise à niveau. Dans le cas contraire, votre moteur pourra être mis à niveau vers une version autre que la plus récente ou que celle à laquelle vous vous attendez.

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à le spécifier avec ce paramètre :

```
  --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de bases de données personnalisé, veillez à le spécifier avec ce paramètre :

```
  ---db-instance-parameter-group-name (name of the custom instance parameter group)
```

Installation d'une mise à niveau du moteur d'une version mineure à l'aide du AWS Management Console

Pour effectuer une mise à niveau d'une version mineure à l'aide de la console Neptune

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le panneau de navigation, choisissez Bases de données, puis choisissez le cluster de bases de données que vous souhaitez modifier.
3. Sélectionnez Modifier.
4. Sous Spécifications de l'instance, choisissez la nouvelle version vers laquelle vous souhaitez procéder à la mise à niveau.
5. Choisissez Suivant.
6. Si vous souhaitez appliquer les modifications immédiatement, choisissez Appliquer immédiatement.
7. Choisissez Soumettre pour mettre à jour votre cluster de base de données.

Installation d'une mise à niveau du moteur d'une version mineure à l'aide du AWS CLI

Vous pouvez utiliser une commande comme la suivante pour effectuer une mise à niveau d'une version mineure sans attendre la fenêtre de maintenance suivante :

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (your-neptune-cluster) \
  --engine-version (new-engine-version) \
  --apply-immediately
```

Si vous effectuez une mise à niveau manuelle à l'aide du AWS CLI, veillez à inclure la version du moteur vers laquelle vous souhaitez effectuer la mise à niveau. Dans le cas contraire, votre moteur

pourra être mis à niveau vers une version autre que la plus récente ou que celle à laquelle vous vous attendez.

Mise à niveau vers la version 1.2.0.0 ou supérieure du moteur à partir d'une version antérieure à 1.2.0.0

La version 1.2.0.0 du moteur inclut plusieurs modifications importantes qui peuvent compliquer plus que d'habitude toute mise à niveau à partir d'une version antérieure :

- La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
- La version 1.2.0.0 du moteur a introduit un nouveau format pour les journaux d'annulation. Par conséquent, si vous effectuez une mise à niveau vers la version 1.2.0.0 ou supérieure à partir d'une version antérieure à 1.2.0.0, la `UndoLogListSize`métrique doit être inférieure à un certain seuil. Dans le cas contraire, le correctif sera annulé et échouera. Les seuils sont basés sur le type d'instance : la limite par défaut est de 40 000 pour les instances 4 x grandes ou plus, et de 10 000 pour les instances inférieures à 4 x large. Si la limite `UndoLogListSize` dépasse la limite lorsque vous tentez de procéder à la mise à niveau, le processus de correction sera annulé, et un événement indiquant le motif sera visible sur la page des événements du cluster. Ces limites peuvent changer pour des raisons opérationnelles sans avertissement préalable.

  Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Le faire avant d'essayer de procéder à la mise à niveau peut aider à abaisser le seuil `UndoLogListSize` en dessous du seuil applicable. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

  Si la `UndoLogListSize` CloudWatch métrique est extrêmement importante, l'ouverture d'un dossier de support peut vous aider à explorer des stratégies supplémentaires pour la ramener en dessous de la limite requise.

- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Utilisation d'un CloudFormation modèle pour mettre à jour la version du moteur de votre cluster de base de données Neptune

Vous pouvez réutiliser le modèle CloudFormation Neptune que vous avez utilisé pour créer votre cluster de base de données Neptune afin de mettre à jour la version de son moteur.

Les mises à niveau des versions du moteur Neptune peuvent être mineures ou majeures. L'utilisation d'un CloudFormation modèle peut faciliter les mises à niveau des versions majeures, qui contiennent souvent des modifications importantes. Comme les mises à niveau de version majeures peuvent contenir des modifications de base de données qui ne sont pas rétrocompatibles avec les applications existantes, vous devrez peut-être également apporter des modifications à vos applications lors de la mise à niveau. Effectuez toujours des tests avant la mise à niveau. Nous vous recommandons aussi vivement de toujours créer un instantané manuel de votre cluster de bases de données avant la mise à niveau.

Notez que vous devez effectuer une mise à niveau du moteur distincte pour chaque version majeure. Vous ne pouvez pas ignorer une version majeure et passer directement à la version majeure suivante.

Avant le 17 mai 2023, si vous utilisiez la CloudFormation pile Neptune pour mettre à niveau la version de votre moteur, elle créait simplement un nouveau cluster de base de données vide à la place de votre cluster actuel. Cependant, depuis le 17 mai 2023, la suite Neptune prend désormais en charge les mises CloudFormation à niveau du moteur sur place qui préservent vos données existantes.

> **Note**
>
> Si vous utilisez le AWS Cloud Development Kit (AWS CDK), assurez-vous que la AWS CDK version utilisée est 2.82.0 ou ultérieure. Les versions antérieures à 2.82.0 ne prennent pas en charge les mises à niveau sur place du moteur Neptune.

Pour une mise à niveau de version majeure, votre modèle doit définir les propriétés suivantes dans `DBCluster` :

- `DBClusterParameterGroup` (personnalisé ou par défaut)
- `DBInstanceParameterGroupName`
- `EngineVersion`

De même, pour DBInstances attached to, DBCluster vous devez définir :

- `DBParameterGroup` (personnalisé/par défaut)

Assurez-vous que tous vos groupes de paramètres sont définis dans le modèle, qu'ils soient par défaut ou personnalisés.

Dans le cas d'un groupe de paramètres personnalisé, assurez-vous que la famille de votre groupe de paramètres personnalisé existant est compatible avec la nouvelle version du moteur. Les versions du moteur antérieures à 1.2.0.0 utilisaient une famille de groupes de paramètres `neptune1`, tandis que les versions du moteur à compter de la version 1.2.0.0 nécessitent une famille de groupes de paramètres `neptune1.2`. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

Pour les mises à niveau majeures des versions du moteur, spécifiez un groupe de paramètres avec la famille appropriée dans le champ `DBInstanceParameterGroupName` de `DBCluster`.

Un groupe de paramètres par défaut doit être mis à niveau vers un groupe compatible avec la nouvelle version du moteur.

Notez que Neptune redémarre automatiquement les instances de base de données après une mise à niveau du moteur.

Rubriques

- Exemple : mise à niveau mineure du moteur de la version 1.2.0.1 vers la version 1.2.0.2
- Exemple : mise à niveau majeure de la version 1.1.1.0 vers 1.2.0.2 avec des groupes de paramètres par défaut
- Exemple : mise à niveau majeure de la version 1.1.1.0 vers 1.2.0.2 avec des groupes de paramètres personnalisés
- Exemple : mise à niveau majeure de la version 1.1.1.0 vers 1.2.0.2 avec une combinaison de groupes de paramètres par défaut et de groupes de paramètres personnalisés

Exemple : mise à niveau mineure du moteur de la version 1.2.0.1 vers la version 1.2.0.2

Recherchez le cluster de bases de données que vous souhaitez mettre à niveau, ainsi que le modèle que vous avez utilisé pour le créer. Par exemple :

```
Description: Base Template to create Neptune Stack with Engine Version 1.2.0.1 using
 custom Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-cluster-parameter-group-description
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-parameter-group-description
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.1
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
```

```
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Mettez à jour la propriété `EngineVersion` en remplaçant `1.2.0.1` par `1.2.0.2` :

```
Description: Template to upgrade minor engine version to 1.2.0.2
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-cluster-parameter-group-description
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-parameter-group-description
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
```

```
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Utilisez-le maintenant CloudFormation pour exécuter le modèle révisé.

Exemple : mise à niveau majeure de la version 1.1.1.0 vers 1.2.0.2 avec des groupes de paramètres par défaut

Recherchez le `DBCluster` que vous souhaitez mettre à niveau, ainsi que le modèle que vous avez utilisé pour le créer. Par exemple :

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 default Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
```

```
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

- Remplacez la valeur `DBClusterParameterGroup` par défaut par celle de la famille de groupes de paramètres utilisée par la nouvelle version du moteur (ici `default.neptune1.2`).
- Pour chaque élément `DBInstance` rattaché au `DBCluster`, remplacez la valeur `DBParameterGroup` par défaut par celle de la famille utilisée par la nouvelle version du moteur (ici `default.neptune1.2`).
- Définissez la propriété `DBInstanceParameterGroupName` sur le groupe de paramètres par défaut de cette famille (ici `default.neptune1.2`).
- Mettez à jour la propriété `EngineVersion` en remplaçant `1.1.0.0` par `1.2.0.2`.

Le modèle devrait se présenter comme suit :

```
Description: Template to upgrade major engine version to 1.2.0.2 by using upgraded
 default parameter groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName: default.neptune1.2
      DBInstanceParameterGroupName: default.neptune1.2
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBCluster
Outputs:
```

```
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
```

Utilisez-le maintenant CloudFormation pour exécuter le modèle révisé.

Exemple : mise à niveau majeure de la version 1.1.1.0 vers 1.2.0.2 avec des groupes de paramètres personnalisés

Recherchez le `DBCluster` que vous souhaitez mettre à niveau, ainsi que le modèle que vous avez utilisé pour le créer. Par exemple :

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 custom Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Name: engineupgradetestcpg
      Family: neptune1
      Description: 'NeptuneDBClusterParameterGroup with family neptune1'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Name: engineupgradetestpg
      Family: neptune1
      Description: 'NeptuneDBParameterGroup1 with family neptune1'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
```

```
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

- Mettez à jour la `DBClusterParameterGroup` famille personnalisée avec celle utilisée par la nouvelle version du moteur `default.neptune1.2` (ici).
- Pour chaque `DBInstance` élément attaché au`DBCluster`, mettez à jour la `DBParameterGroup` famille personnalisée en fonction de celle utilisée par la nouvelle version du moteur (ici`default.neptune1.2`).
- Définissez la propriété `DBInstanceParameterGroupName` sur le groupe de paramètres de cette famille (ici `default.neptune1.2`).
- Mettez à jour la propriété `EngineVersion` en remplaçant `1.1.0.0` par `1.2.0.2`.

Le modèle devrait se présenter comme suit :

```
Description: Template to upgrade major engine version to 1.2.0.2 by modifying existing
 custom parameter groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
```

```
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Name: engineupgradetestcpgnew
      Family: neptune1.2
      Description: 'NeptuneDBClusterParameterGroup with family neptune1.2'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Name: engineupgradetestpgnew
      Family: neptune1.2
      Description: 'NeptuneDBParameterGroup1 with family neptune1.2'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
      DBInstanceParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Utilisez-le maintenant CloudFormation pour exécuter le modèle révisé.

Exemple : mise à niveau majeure de la version 1.1.1.0 vers 1.2.0.2 avec une combinaison de groupes de paramètres par défaut et de groupes de paramètres personnalisés

Recherchez le `DBCluster` que vous souhaitez mettre à niveau, ainsi que le modèle que vous avez utilisé pour le créer. Par exemple :

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 custom Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1
      Description: 'NeptuneDBClusterParameterGroup with family neptune1'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1
      Description: 'NeptuneDBParameterGroup with family neptune1'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  CustomNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
```

```
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
  DefaultNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

- Pour un groupe de paramètres de cluster personnalisé, mettez à jour la famille `DBClusterParameterGroup` en utilisant celle correspondant à la nouvelle version du moteur, à savoir `neptune1.2`.
- Pour un groupe de paramètres de cluster par défaut, mettez à jour la valeur `DBClusterParameterGroup` en utilisant la valeur par défaut correspondant à la nouvelle version du moteur, à savoir `default.neptune1.2`.
- Pour chaque élément `DBInstance` attaché au `DBCluster`, remplacez un élément `DBParameterGroup` par défaut par celui de la famille utilisée par la nouvelle version du moteur (ici `default.neptune1.2`), et un groupe de paramètres personnalisé par celui qui utilise la famille prise en charge par la nouvelle version du moteur (ici `neptune1.2`).
- Définissez la propriété `DBInstanceParameterGroupName` sur le groupe de paramètres de la famille, pris en charge par la nouvelle version du moteur.

Le modèle devrait se présenter comme suit :

```
Description: Template to update Neptune Stack to Engine Version 1.2.0.1 using custom
 and default Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1.2
      Description: 'NeptuneDBClusterParameterGroup with family neptune1.2'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: 'NeptuneDBParameterGroup1 with family neptune1.2'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
      DBInstanceParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBClusterParameterGroup
  CustomNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
```

```
  DefaultNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Utilisez-le maintenant CloudFormation pour exécuter le modèle révisé.

Clonage de bases de données dans Neptune

Le clonage de base de données vous permet de créer de façon rapide et économique des clones de toutes vos bases de données dans Amazon Neptune. Les bases de données clone n'ont besoin que d'un espace supplémentaire minime au moment où elles sont créées. Le clonage de base de données utilise un copy-on-write protocole. Les données sont copiées au moment où elles sont modifiées, que ce soit dans les bases de données source ou les bases de données clone. Vous pouvez créer plusieurs clones du même cluster de base de données. Vous pouvez également créer des clones supplémentaires à partir d'autres clones. Pour plus d'informations sur le fonctionnement copy-on-write du protocole dans le contexte du stockage Neptune, consultez. Copy-on-Write Protocole

Vous pouvez utiliser le clonage de base de données dans divers cas d'utilisation, notamment lorsque vous ne voulez pas affecter votre environnement de production, comme dans les exemples suivants :

- Expérimentation et évaluation de l'impact de modifications, telles que des modifications du schéma ou des modifications du groupe de paramètres.
- Réalisation d'opérations imposant une charge de travail élevée, telles que l'exportation de données ou l'exécution de requêtes analytiques.
- Création d'une copie d'un cluster de base de données de production dans un environnement autre que de production pour le développement ou les tests.

Pour créer un clone d'un cluster de base de données à l'aide du AWS Management Console

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le panneau de navigation, choisissez Instances. Choisissez l'instance principale pour le cluster de base de données dont vous voulez créer un clone.
3. Choisissez Instance actions (Actions d'instance), puis Create clone (Créer un clone).
4. Dans la page Create Clone (Créer un clone), entrez un nom pour l'instance principale du cluster de base de données clone dans le champ DB instance identifier (Identifiant d'instance DB).

   Configurez éventuellement d'autres paramètres pour le cluster de base de données clone. Pour plus d'informations sur les différents paramètres de cluster de base de données, consultez Lancement à l'aide de la console.
5. Choisissez Create Clone (Créer un clone) pour lancer le cluster de base de données clone.

Pour créer un clone d'un cluster de base de données à l'aide du AWS CLI

- Appelez la point-in-time AWS CLI commande Neptune restore-db-cluster-to- et saisissez les valeurs suivantes :
  - `--source-db-cluster-identifier` : nom du cluster de bases de données source à partir duquel un clone doit être créé.
  - `--db-cluster-identifier` : nom du cluster de bases de données clone.
  - `--restore-type copy-on-write` : la valeur `copy-on-write` indique qu'un cluster de bases de données clone doit être créé.
  - `--use-latest-restorable-time` : précise que la dernière heure de sauvegarde restaurable doit être utilisée.

> **Note**
>
> La point-in-time AWS CLI commande restore-db-cluster-to- clone uniquement le cluster de base de données, et non les instances de base de données pour ce cluster de base de données.

L' Linux/UNIX exemple suivant crée un clone à partir du `source-db-cluster-id` cluster de base de données et le nommé`db-clone-cluster-id`.

```
aws neptune restore-db-cluster-to-point-in-time \
  --region us-east-1 \
  --source-db-cluster-identifier source-db-cluster-id \
  --db-cluster-identifier db-clone-cluster-id \
  --restore-type copy-on-write \
  --use-latest-restorable-time
```

Le même exemple fonctionne sur Windows si le caractère d'échappement de fin de ligne \ est remplacé par l'équivalent Windows ^ :

```
aws neptune restore-db-cluster-to-point-in-time ^
  --region us-east-1 ^
  --source-db-cluster-identifier source-db-cluster-id ^
  --db-cluster-identifier db-clone-cluster-id ^
  --restore-type copy-on-write ^
```

```
  --use-latest-restorable-time
```

Limites

Le clonage de base de données dans Neptune présente les limites suivantes :

- Vous ne pouvez pas créer de bases de données AWS clonées dans différentes régions. Les bases de données clone doivent être créées dans la même région que les bases de données source.
- Une base de données clonée utilise toujours le correctif le plus récent de la version du moteur Neptune utilisée par la base de données à partir de laquelle elle a été clonée. Ceci est vrai même si la base de données source n'a pas encore été mise à niveau vers cette version de correctif. Cependant, la version du moteur elle-même ne change pas.
- Actuellement, vous êtes limité à 15 clones par copie du cluster de bases de données Neptune, ce qui comprend les clones basés sur d'autres clones. Une fois cette limite atteinte, vous devez effectuer une autre copie de votre base de données plutôt que de la cloner. Cependant, si vous créez une autre copie, elle peut aussi avoir jusqu'à 15 clones.
- Le clonage de base de données entre comptes n'est pas pris en charge actuellement.
- Vous pouvez fournir un réseau Virtual Private Cloud (VPC) différent pour votre clone. Toutefois, les sous-réseaux qu'ils contiennent VPCs doivent correspondre au même ensemble de zones de disponibilité.

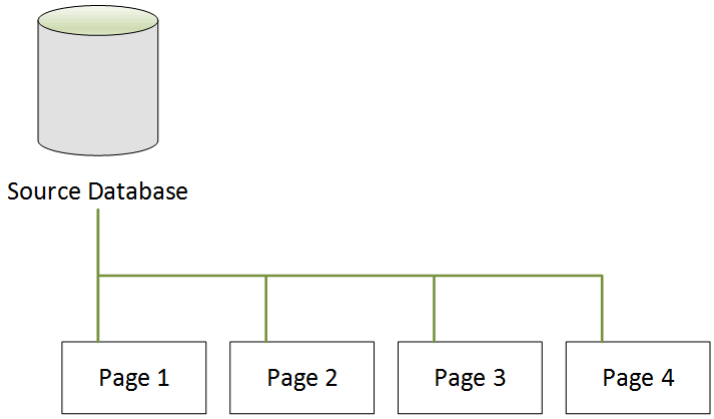
Copy-on-Write Protocole pour le clonage de bases de données

Les scénarios suivants illustrent le fonctionnement copy-on-write du protocole.

- Base de données Neptune avant le clonage
- Base de données Neptune après le clonage
- Lorsqu'une modification est apportée à la base de données source
- Lorsqu'une modification est apportée à la base de données clone

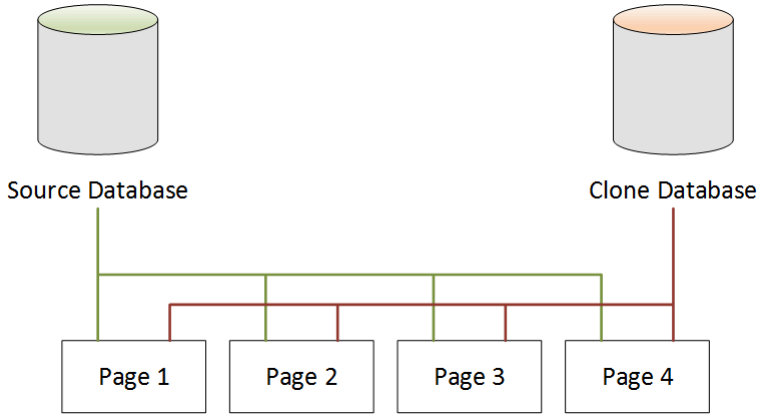
Base de données Neptune avant le clonage

Les données figurant dans une base de données source sont stockées dans des pages. Dans le schéma ci-dessous, la base de données source comporte quatre pages.



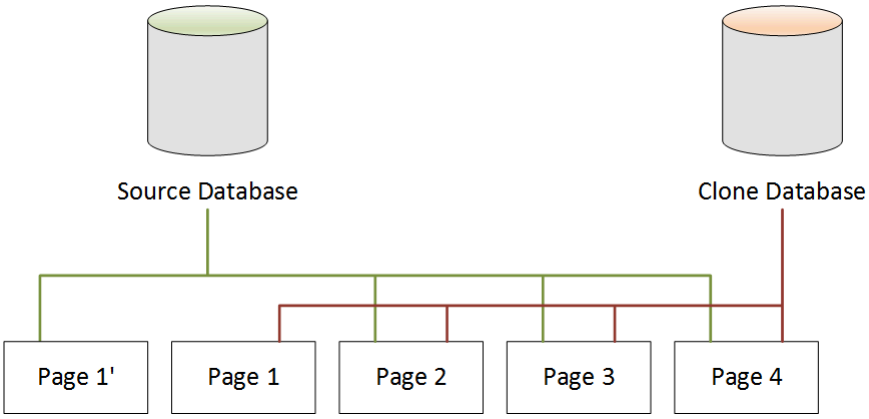
Base de données Neptune après le clonage

Comme le montre le schéma suivant, aucune modification n'a été apportée à la base de données source après le clonage de base de données. La base de données source et la base de données clone pointent toutes les deux sur les quatre mêmes pages. Aucune page n'a été copiée physiquement et aucun stockage supplémentaire n'est nécessaire.



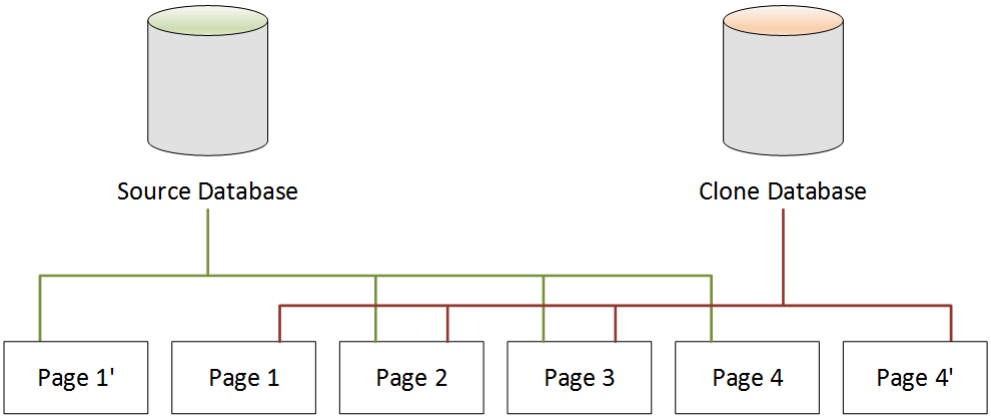
Lorsqu'une modification est apportée à la base de données source

Dans l'exemple suivant, la base de données source apporte une modification aux données de la `Page 1`. Au lieu d'écrire dans la `Page 1` d'origine, il utilise un stockage supplémentaire pour créer une nouvelle page, appelée `Page 1'`. La base de données source pointe maintenant sur la nouvelle `Page 1'`, ainsi que sur la `Page 2`, la `Page 3` et la `Page 4`. La base de données clone continue de pointer sur les `Page 1` à `Page 4`.



Lorsqu'une modification est apportée à la base de données clone

Dans le schéma ci-dessous, la base de données clone a également été modifiée, cette fois au niveau de la `Page 4`. Au lieu d'écrire dans la page `Page 4` d'origine, un stockage supplémentaire est utilisé pour créer une nouvelle page, appelée `Page 4'`. La base de données source continue à pointer vers la `Page 1'`, ainsi que vers les pages `Page 2` à `Page 4`, mais la base de données clone pointe maintenant vers les pages `Page 1` à `Page 3`, ainsi que vers la `Page 4'`.



Comme le montre le deuxième scénario, après le clonage de base de données, aucun stockage supplémentaire n'est nécessaire au stade de la création du clone. Toutefois, lorsque des modifications interviennent dans la base de données source et la base de données clone, comme le montrent les troisième et quatrième scénarios, seules les pages modifiées sont créées. Lorsque des modifications supplémentaires interviennent au fil du temps dans la base de données source et la base de données clone, vous avez besoin de plus de stockage pour capturer et stocker ces modifications.

Suppression d'une base de données source

La suppression d'une base de données source n'affecte pas les bases de données clone qui lui sont associées. Les bases de données clones continuent de pointer sur les pages qui étaient précédemment la propriété de la base de données source.

Gestion des instances Amazon Neptune

Les sections suivantes fournissent des informations sur les opérations réalisées au niveau de l'instance.

Rubriques

- Classe d'instances extensible T3 Neptune
- Modification d'une instance de base de données Neptune (et application immédiate)
- Changement de nom d'une instance de base de données Neptune
- Redémarrage d'une instance de base de données dans Amazon Neptune
- Suppression d'une instance de base de données dans Amazon Neptune

Classe d'instances extensible T3 Neptune

En plus des classes d'instances à performances fixes telles que R5 et R6, Amazon Neptune offre la possibilité d'utiliser une instance T3 à performances extensibles. Lorsque vous développez votre application orientée graphe, il est important que votre base de données soit rapide et réactive, mais vous n'avez pas besoin de l'utiliser en permanence. La classe d'instances db.t3.medium de Neptune est exactement ce que vous devez utiliser dans ce cas, à un coût nettement inférieur à celui de la classe d'instances à performances fixes la moins chère.

Une instance extensible fonctionne à un niveau de base de performances de l'UC. Lorsqu'une charge de travail demande davantage, l'instance dépasse largement ce niveau de base aussi longtemps que nécessaire. Son prix horaire couvre les pics, à condition que l'utilisation moyenne de l'UC ne dépasse pas le niveau de base sur une période de 24 heures. Pour la plupart des situations de développement et de test, cela se traduit par de bonnes performances à faible coût.

Si vous commencez par une classe d'T3instance, vous pouvez facilement passer ultérieurement à une classe d'instance à performances fixes lorsque vous serez prêt à passer en production, en utilisant le AWS Management Console AWS CLI, ou l'un des. AWS SDKs

L'extension T3 est régie par les crédits d'UC

Un crédit d'UC représente l'utilisation complète d'un cœur d'UC virtuelle (vCPU) pendant une minute. Cela peut également se traduire par une utilisation de 50 % d'un vCPU pendant deux minutes, ou une utilisation de 25 % de deux vCPU CPUs pendant deux minutes, etc.

Une instance T3 accumule des crédits d'UC lorsqu'elle est inactive et les utilise lorsqu'elle est active, selon une résolution de mesures en millisecondes. La classe d'db.t3.mediuminstance possède deux vCPUs, dont chacun rapporte 12 crédits CPU par heure lorsqu'il est inactif. Cela signifie que 20 % d'utilisation de chaque UC virtuelle se traduit par un solde égal à zéro UC. Les 12 crédits d'UC gagnés sont dépensés par les 20 % d'utilisation de l'UC virtuelle (20 % de 60 minutes étant égal à 12). Cette utilisation de 20 % correspond donc au taux d'utilisation de base qui produit un solde de crédit d'UC ni positif ni négatif.

Le temps d'inactivité (utilisation de l'UC inférieure à 20 % du total disponible) entraîne le stockage des crédits d'UC dans un compartiment de solde de crédit, jusqu'à la limite de 576 pour une classe d'instance db.t3.medium(nombre maximal de crédits d'UC pouvant être accumulés en 24 heures, soit 2 x 12 x 24). Au-delà de cette limite, les crédits d'UC sont simplement éliminés.

Si nécessaire, l'utilisation de l'UC peut atteindre 100 % aussi longtemps que nécessaire pour une charge de travail, même si le solde de crédits d'UC passe en dessous de zéro. Si l'instance conserve

un solde négatif en continu pendant 24 heures, des frais supplémentaires équivalents à 0,05 USD par tranche de -60 crédits d'UC accumulés au cours de cette période seront facturés. Toutefois, pour la plupart des charges de travail de développement et de test, les pics sont généralement couverts par les temps d'inactivité avant ou après.

> **Note**
>
> La classe d'T3instance de Neptune est configurée comme le mode Amazon EC2 unlimited.

Utilisation du AWS Management Console pour créer une instance T3 Burstable

Dans le AWS Management Console, vous pouvez créer une instance de cluster de base de données principale ou une instance en lecture qui utilise la classe d'`db.t3.medium`instance, ou vous pouvez modifier une instance existante pour utiliser la classe d'`db.t3.medium`instance.

Par exemple, pour créer une instance principale de cluster de bases de données dans la console Neptune :

- Choisissez Create database (Créer une base de données).
- Choisissez une version de moteur de base de données égale ou ultérieure à `1.0.2.2`.
- Sous Purpose (Objet), choisissez Development and Testing (Développement et Test).
- Pour DB instance class (Classe d'instance de base de données), acceptez la valeur par défaut : `db.t3.medium — 2 vCPU, 4 GiB RAM`.

Utilisation du AWS CLI pour créer une instance T3 Burstable

Vous pouvez également utiliser le AWS CLI pour faire la même chose :

```
aws neptune create-db-cluster \
    --db-cluster-identifier (name for a new DB cluster) \
    --engine neptune \
    --engine-version "1.0.2.2"

aws neptune create-db-instance \
    --db-cluster-identifier (name of the new DB cluster) \
    --db-instance-identifier (name for the primary writer instance in the cluster) \
    --engine neptune \
    --db-instance-class db.t3.medium
```

Modification d'une instance de base de données Neptune (et application immédiate)

Vous pouvez appliquer la plupart des modifications apportées à une instance de base de données Amazon Neptune immédiatement ou les reporter à la fenêtre de maintenance suivante. Certaines modifications, telles que les modifications d'un groupe de paramètres, nécessitent que vous redémarriez manuellement votre instance de base de données pour que la modification entre en vigueur.

> **Important**
>
> Les modifications entraînent une interruption si Neptune doit redémarrer votre instance de base de données pour qu'elles soient appliquées. Analysez l'impact sur votre base de données et les applications avant de modifier vos paramètres d'instance de base de données.

Paramètres courants et implications sur les temps d'arrêt.

Le tableau suivant contient des détails sur les paramètres que vous pouvez modifier, et indique quand les modifications peuvent être appliquées et si les modifications entraînent un temps d'arrêt pour l'instance de base de données.

Paramètres de l'instance de base de données	Remarques sur la durée d'indisponibilité	
Classe d'instances de base de données	Une interruption a lieu lors de cette modification, qu'elle soit appliquée immédiate ment ou lors de la fenêtre de maintenance suivante.	
Identifiant d'instance de base de données	L'instance de base de données est redémarrée, et une interruption a lieu lors de cette modification, qu'elle soit appliquée immédiate	

Paramètres de l'instance de base de données	Remarques sur la durée d'indisponibilité	
	ment ou lors de la fenêtre de maintenance suivante.	
Groupe de sous-réseaux	L'instance de base de données est redémarrée, et une interruption a lieu lors de cette modification, qu'elle soit appliquée immédiate ment ou lors de la fenêtre de maintenance suivante.	
Groupe de sécurité	La modification est appliquée de manière asynchrone dès que possible, quelle que soit la date à laquelle vous spécifiez que les modifications doivent avoir lieu, et aucune interrupt ion n'en résulte.	–
Autorité de certification	Par défaut, l'instance de base de données est redémarré e lorsque vous attribuez une nouvelle autorité de certifica tion.	
Database Port	La modification se produit toujours immédiatement, ce qui entraîne le redémarra ge de l'instance de base de données et une interruption.	

Paramètres de l'instance de base de données	Remarques sur la durée d'indisponibilité	
Groupe de paramètres de base de données	La modification de ce paramètre n'entraîne pas d'interruption. Le nom du groupe de paramètres lui-même est immédiatement changé, mais les modifications réelles des paramètres ne prennent effet qu'après le redémarrage de l'instance, sans basculement. Dans ce cas, l'instance de base de données n'est pas automatiquement redémarrée, et les modifications de paramètre ne sont pas appliquées pendant la fenêtre de maintenance suivante. Toutefois, si vous modifiez des paramètres dynamiques dans le groupe de paramètres de base de données nouvellement associé, ces modifications sont appliquées immédiatement sans redémarrage. Pour de plus amples informations, veuillez consulter Redémarrage d'une instance de base de données dans Amazon Neptune.	

Paramètres de l'instance de base de données	Remarques sur la durée d'indisponibilité	
Groupe de paramètres de cluster de bases de données	La modification du groupe de paramètres de base de données a lieu immédiate ment.	
Période de rétention des sauvegardes	Si vous spécifiez que les modifications doivent interveni r immédiatement, cette modification a lieu immédiate ment. Dans le cas contraire, si vous remplacez le paramètre d'une valeur non nulle par une autre valeur non nulle, la modification est appliquée de manière asynchrone dès que possible. Toute autre modificat ion sera appliquée pendant la fenêtre de maintenance suivante. Une interruption se produit si vous passez de 0 à une valeur non nulle, ou d'une valeur non nulle à 0.	

Paramètres de l'instance de base de données	Remarques sur la durée d’indisponibilité	
Journal d'audit	Sélectionnez Journal d'audit si vous souhaitez utiliser la journalisation d'audit via CloudWatch Logs. Vous devez également définir le paramètre `neptune_enable_audit_log` du groupe de paramètres du cluster de bases de données sur `enable` (1) pour que la journalisation des audits soit activée.	
Mise à niveau automatique de versions mineures	Sélectionnez Activer la mise à niveau automatique des versions mineures si vous souhaitez que votre cluster de bases de données Neptune reçoive automatiquement les mises à niveau des versions mineures lors de leur publication. L'option Mise à niveau automatique des versions mineures s'applique uniquement aux mises à niveau des versions mineures pour votre cluster de bases de données Amazon Neptune. Elle ne s’applique pas aux correctifs réguliers appliqués pour maintenir la stabilité du système.	

Changement de nom d'une instance de base de données Neptune

Vous pouvez renommer une instance de base de données Amazon Neptune à l'aide de la AWS Management Console. Renommer une instance de base de données peut avoir des effets à grande portée. La liste suivante correspond à ce que vous devez savoir avant de renommer une instance de base de données.

- Lorsque vous renommez une instance de base de données, le point de terminaison de cette dernière change, parce que l'URL inclut le nom que vous avez attribué à l'instance de base de données. Vous devez toujours rediriger le trafic de l'ancienne URL vers la nouvelle.
- Lorsque vous renommez une instance de base de données, l'ancien nom DNS qui a été utilisé par l'instance de base de données est immédiatement supprimé, mais peut rester dans le cache quelques minutes. Le nouveau nom DNS de l'instance de base de données renommée devient effectif après environ 10 minutes. L'instance de base de données renommée n'est pas disponible jusqu'à ce que le nouveau nom ne devienne effectif.
- Vous ne pouvez pas utiliser un nom d'instance de base de données existant lorsque vous renommez une instance.
- Tous les réplicas en lecture associés à une instance de base de données demeurent associés à cette instance une fois qu'elle a été renommée. Supposons par exemple que vous ayez une instance de base de données qui desserve votre base de données de production et que l'instance ait plusieurs réplicas en lecture associés. Si vous renommez l'instance de base de données, puis la remplacez dans l'environnement de production par un instantané de base de données, l'instance de base de données que vous avez renommée conserve les réplicas en lecture associés.
- Les métriques et les événements associés au nom d'une instance de base de données sont conservés si vous réutilisez un nom d'instance de base de données. Par exemple, si vous effectuez la promotion d'un réplica en lecture et que vous remplacez son nom par celui de l'instance de base de données principale précédente, les événements et les métriques qui étaient associés à l'instance de base de données principale sont associés à l'instance renommée.
- Les balises de l'instance de base de données demeurent avec l'instance de base de données, quel que soit le changement de nom.
- Les snapshots DB sont conservés pour une instance de base de données renommée.

Pour renommer une instance de base de données à l'aide de la console Neptune

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.

2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Cochez la case en regard de l'instance de base de données que vous voulez renommer.
4. Dans le menu Instance Actions (Actions sur l'instance), sélectionnez Modify (Modifier).
5. Saisissez un nouveau nom dans la zone de texte DB Instance Identifier (Identifiant d’instance de base de données). Sélectionnez Apply immediately (Appliquer immédiatement), puis choisissez Continue (Continuer).
6. Choisissez Modify DB instance (Modifier l'instance de base de données) pour terminer la modification.

Redémarrage d'une instance de base de données dans Amazon Neptune

Dans certains cas, si vous modifiez une instance de base de données Amazon Neptune, le groupe de paramètres de base de données associé à l'instance ou un paramètre de base de données statique dans un groupe de paramètres que l'instance utilise, vous devez redémarrer l'instance pour que ces modifications soient appliquées.

Le redémarrage d'une instance de base de données entraîne celui du service du moteur de base de données. Un redémarrage applique également à l'instance de base de données toutes les modifications du groupe de paramètres de base de données associé qui étaient en attente. Le redémarrage d'une instance de base de données entraîne une interruption momentanée de l'instance, au cours de laquelle le statut de l'instance de base de données est défini sur rebooting. Si l'instance Neptune est configurée pour les déploiements multi-AZ, le redémarrage peut être effectué avec un basculement. Un événement Neptune est créé lorsque le redémarrage est terminé.

Si votre instance de base de données est un déploiement Multi-AZ, vous pouvez forcer un basculement d'une zone de disponibilité vers une autre lorsque vous choisissez l'option Reboot (Redémarrer). Lorsque vous forcez un basculement de votre instance de base de données, Neptune bascule automatiquement vers un réplica de secours dans une autre zone de disponibilité. Ensuite, elle met à jour l'enregistrement DNS pour que l'instance de base de données pointe vers l'instance de base de données en veille. Par conséquent, vous devez nettoyer et rétablir toutes les connexions existantes à votre instance de bases de données.

Reboot with failover (Redémarrer avec basculement) présente un avantage lorsque vous souhaitez simuler une défaillance d'une instance de base de données à des fins de test, ou restaurer la zone de disponibilité d'origine des opérations après un basculement. Pour en savoir plus, consultez la section Haute disponibilité (Multi-AZ) dans le Guide de l'utilisateur Amazon RDS. Lorsque vous redémarrez un cluster de base de données, il bascule vers le réplica de secours. Le redémarrage d'un réplica Neptune ne déclenche pas de basculement.

Le temps nécessaire au redémarrage est une fonction du processus de récupération sur incident. Pour améliorer le délai de redémarrage, nous vous recommandons de réduire les activités de bases de données autant que possible pendant le processus de redémarrage pour réduire l'activité de restauration pour les transactions en transit.

Sur la console, l'option Reboot (Redémarrer) peut être désactivée si l'instance de base de données n'est pas dans l'état Available (Disponible). Cela peut être dû à plusieurs raisons, par exemple une sauvegarde en cours, une modification demandée par le client ou encore une action du créneau de maintenance.

> **Note**
>
> Avant la Sortie : 1.2.0.0 (21/07/2022), toutes les instances de réplica en lecture d'un cluster de bases de données étaient automatiquement redémarrées lors du redémarrage de l'instance (d'enregistreur) principale.
> À compter de la Sortie : 1.2.0.0 (21/07/2022), le redémarrage de l'instance principale n'entraîne le redémarrage d'aucun des réplicas. Dès lors, si vous modifiez un paramètre du cluster, vous devez redémarrer chaque instance séparément pour que cette modification de paramètre s'applique (voir Groupes de paramètres).

Pour redémarrer une instance de base de données à l'aide de la console Neptune

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez l'instance de base de données que vous souhaitez redémarrer.
4. Choisissez Actions d'instance, puis Redémarrer.
5. Pour forcer un basculement d'une zone de disponibilité à une autre, sélectionnez Reboot with failover? (Redémarrer avec basculement ?) dans la boîte de dialogue Reboot DB Instance (Redémarrer l'instance de base de données).
6. Choisissez Redémarrer. Pour annuler le redémarrage, choisissez Cancel (Annuler).

Suppression d'une instance de base de données dans Amazon Neptune

Vous pouvez supprimer une instance de base de données Amazon Neptune dans n'importe quel état et à tout moment, tant que l'instance a été démarrée.

> **Warning**
>
> Si vous supprimez la dernière instance restante d'un cluster à l'aide de la console Web, le volume de stockage du cluster sous-jacent sera également supprimé.

Prise d'un instantané final de votre instance de base de données avant de la supprimer

Pour supprimer une instance de base de données, vous devez spécifier le nom de l'instance et préciser si vous voulez qu'il soit pris un instantané de base de données final de l'instance. Si l'instance de base de données que vous supprimez a le statut Creating, il n'est pas possible de prendre un instantané de base de données final. Si l'instance de base de données est dans un état d'échec avec le statut failed, incompatible-restore ou incompatible-network, vous pouvez uniquement supprimer l'instance quand le paramètre `SkipFinalSnapshot` a la valeur `true`.

Si vous supprimez toutes les instances de base de données Neptune d'un cluster de base de données à l'aide de AWS Management Console, l'ensemble du cluster de base de données est automatiquement supprimé. Si vous utilisez le SDK AWS CLI ou le SDK, vous devez supprimer le cluster de base de données manuellement après avoir supprimé la dernière instance.

> **Important**
>
> Si vous supprimez un cluster de bases de données entier, toutes ses sauvegardes automatiques sont supprimées en même temps et ne peuvent pas être récupérées. En d'autres termes, si vous ne choisissez pas de créer manuellement un instantané de bases de données final, vous ne pourrez pas restaurer l'instance de base de données à son état final ultérieurement. Les instantanés manuels d'une instance ne sont pas supprimés lorsque vous supprimez un cluster.

Si l'instance de base de données que vous souhaitez supprimer possède un réplica en lecture, vous devez promouvoir ce dernier ou le supprimer.

Dans les exemples suivants, vous supprimez une instance de base de données avec et sans snapshot DB final.

Suppression d'une instance de base de données sans snapshot final

Si vous voulez supprimer rapidement une instance de base de données, vous pouvez ignorer la création d'un instantané de base de données final. Lorsque vous supprimez une instance de base de données, toutes les sauvegardes automatiques sont supprimées et ne peuvent pas être récupérées. Les instantanés manuels ne sont pas supprimés.

Pour supprimer une instance de base de données sans instantané de base de données final à l'aide de la console Neptune

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Dans la liste Instances, sélectionnez la case d'option en regard de l'instance de base de données que vous souhaitez supprimer.
4. Choisissez Actions d'instance), puis Supprimer.
5. Choisissez Non dans la case Créer un instantané final ?.
6. Sélectionnez Delete (Supprimer).

Suppression d'une instance de base de données avec snapshot final

Si vous voulez pouvoir restaurer ultérieurement une instance de base de données supprimée, vous pouvez créer un instantané de base de données final. Toutes les sauvegardes automatiques sont également supprimées et ne peuvent pas être récupérées. Les instantanés manuels ne sont pas supprimés.

Pour supprimer une instance de base de données avec un instantané de base de données final à l'aide de la console Neptune

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Dans la liste Instances, sélectionnez la case d'option en regard de l'instance de base de données que vous souhaitez supprimer.

4. Choisissez Actions d'instance), puis Supprimer.
5. Choisissez Oui dans la case Créer un instantané final ?.
6. Dans la zone Nom de l'instantané final, saisissez le nom de votre instantané de base de données final.
7. Sélectionnez Supprimer.

Vous pouvez vérifier l'état de santé d'une instance, déterminer son type, identifier la version du moteur actuellement installée et obtenir d'autres informations sur une instance à l'aide de l'API instance-status.

Capture des modifications de graphe en temps réel à l'aide des flux Neptune

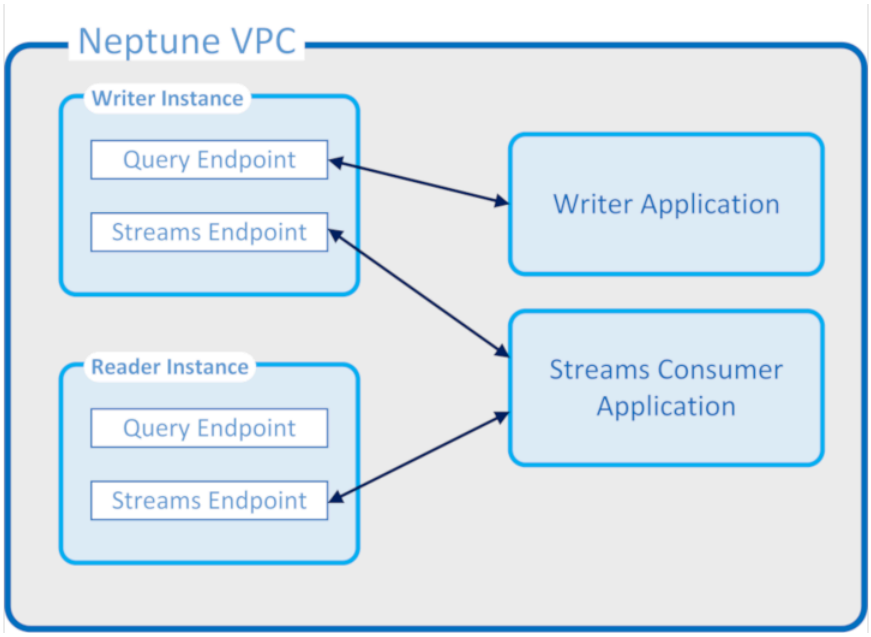
Neptune Streams journalise toutes les modifications apportées à votre graphe au fur et à mesure qu'elles sont appliquées, dans l'ordre dans lequel elles sont effectuées, de manière entièrement gérée. Une fois que Streams est activé, Neptune s'occupe de la disponibilité, de la sauvegarde, de la sécurité et de l'expiration des données.

Voici quelques-uns des nombreux cas d'utilisation dans lesquels vous pouvez capturer les modifications apportées à un graphe au fur et à mesure de leur application :

- Vous pouvez souhaiter que votre application informe automatiquement les personnes concernées lorsque certaines modifications sont apportées.
- Vous souhaiterez peut-être également conserver une version actuelle de vos données graphiques dans un autre magasin de données, tel qu'Amazon OpenSearch Service ElastiCache, Amazon ou Amazon Simple Storage Service (Amazon S3).

Neptune utilise le même stockage natif pour le flux de journaux des modifications que pour les données de graphe. Il écrit les entrées de journal des modifications de façon synchronisée avec la transaction qui effectue ces modifications. Vous récupérez ces enregistrements de modification à partir du flux de journaux à l'aide d'une API REST HTTP. (Pour de plus amples informations, veuillez consulter Appel de l'API Streams.)

Le schéma suivant montre comment les données des journaux de modifications peuvent être récupérées à partir de Neptune Streams.



Garanties des flux Neptune

- Les modifications apportées par une transaction sont immédiatement disponibles en lecture auprès de l'enregistreur et des lecteurs dès que la transaction est terminée (en l'absence de retard de réplication normal au niveau des lecteurs).
- Les enregistrements de modification apparaissent strictement séquentiellement, dans l'ordre dans lequel ils se sont produits (cela inclut les modifications apportées au sein d'une transaction).
- Les flux de modifications ne contiennent aucun doublon. Chaque modification n'est consignée qu'une seule fois.
- Les flux de modifications sont arrivés au terme de leur exécution. Aucune modification n'est perdue ou omise.
- Les flux de modifications contiennent toutes les informations nécessaires pour déterminer l'état complet de la base de données elle-même à tout moment, à condition que l'état de départ soit connu.
- La fonction Streams peut être activée ou désactivée à tout moment.

Propriétés opérationnelles des flux Neptune

- Le flux de journaux des modifications est entièrement géré.
- Change-log les données sont écrites de manière synchrone dans le cadre de la même transaction qui apporte une modification.

- Lorsque Neptune Streams est activé, vous devez payer des frais I/O de stockage associés aux données du journal des modifications.
- Par défaut, les enregistrements de modifications sont purgés automatiquement une semaine après leur création. À partir de la version 1.2.0.0 du moteur, cette période de rétention peut être remplacée à l'aide du paramètre de cluster de bases de données neptune_streams_expiry_days par un nombre quelconque de jours compris entre 1 et 90.
- Les performances de lecture sur les flux évoluent avec les instances.
- Vous pouvez obtenir une haute disponibilité et un débit de lecture élevé à l'aide des réplicas en lecture. Il n'y a pas de limite au nombre de lecteurs de flux que vous pouvez créer et utiliser simultanément.
- Change-log les données sont répliquées sur plusieurs zones de disponibilité, ce qui les rend extrêmement durables.
- Les données de journal sont aussi sécurisées que vos données de graphe. Elles peuvent être chiffrées au repos et en transit. L'accès peut être contrôlé à l'aide d'IAM, d'Amazon VPC AWS Key Management Service et AWS KMS(). Tout comme les données de graphe, les données de journal peuvent être sauvegardées et restaurées ultérieurement à l'aide de restaurations à un instant donné (PITR).
- L'écriture synchrone des données de flux dans le cadre de chaque transaction entraîne une légère dégradation des performances d'écriture globales.
- Les données de flux ne sont pas partitionnées, car Neptune ne comporte qu'une seule partition, de par sa conception.
- L'API `GetRecords` du flux de journaux utilise les mêmes ressources que toutes les autres opérations de graphe Neptune. Cela signifie que les clients doivent équilibrer la charge entre les demandes de flux et les autres demandes de la base de données.
- Lorsque la fonction Streams est désactivée, toutes les données des journaux deviennent immédiatement inaccessibles. Cela signifie que vous devez lire toutes les données de journaux qui vous intéressent avant de désactiver la journalisation.
- Il n'existe actuellement aucune intégration native avec AWS Lambda. Le flux de journaux ne génère pas d'événement qui puisse déclencher une fonction Lambda.

Rubriques

- Utilisation de Neptune Streams
- Formats de sérialisation dans Neptune Streams
- Exemples Neptune Streams

- Utilisation AWS CloudFormation pour configurer la Neptune-to-Neptune réplication avec l'application Streams Consumer
- Utilisation de la réplication des flux Neptune entre régions pour la reprise après sinistre

Utilisation de Neptune Streams

La fonctionnalité Neptune Streams vous permet de générer une séquence complète d'entrées de journal des modifications qui enregistrent chaque modification apportée aux données de votre graphe au fur et à mesure qu'elles sont appliquées. Pour obtenir une présentation de cette fonction, veuillez consulter Capture des modifications de graphe en temps réel à l'aide des flux Neptune.

Rubriques

- Activation de Neptune Streams
- Désactivation de Neptune Streams
- Appel de l'API REST Neptune Streams
- Format de réponse de l'API Neptune Streams
- Lire les données du flux Neptune
- Exceptions de l'API Neptune Streams

Activation de Neptune Streams

Vous pouvez activer ou désactiver Neptune Streams à tout moment en définissant le paramètre de cluster de bases de données `neptune_streams`. Si le paramètre est défini sur `1`, Streams est activé, et s'il est défini sur `0`, Streams est désactivé.

> **Note**
>
> Après avoir modifié le paramètre `neptune_streams` du cluster de bases de données, vous devez redémarrer toutes les instances de base de données du cluster pour que la modification soit effective.

Vous pouvez définir le paramètre de cluster de bases de données neptune_streams_expiry_days pour contrôler le nombre de jours (compris entre 1 et 90) pendant lesquels les enregistrements de flux resteront sur le serveur avant d'être supprimés. La valeur par défaut est 7.

Initialement, Neptune Streams a été ajouté en tant que fonctionnalité expérimentale que vous activiez ou désactiviez en mode laboratoire à l'aide du paramètre `neptune_lab_mode` du cluster de bases de données (voir Mode expérimental Neptune). L'utilisation du mode Lab pour activer Streams est désormais obsolète et sera désactivée à l'avenir.

Désactivation de Neptune Streams

Vous pouvez désactiver Neptune Streams à tout moment.

Pour désactiver Streams, mettez à jour le groupe de paramètres de cluster de bases de données afin que la valeur du paramètre `neptune_streams` soit définie sur 0.

> **Important**
>
> Dès que Streams est désactivé, vous ne pouvez plus accéder aux données du journal des modifications. Veillez à lire les informations qui vous intéressent avant de désactiver Streams.

Appel de l'API REST Neptune Streams

Vous accédez à Neptune Streams à l'aide d'une API REST qui envoie une demande HTTP GET à l'un des points de terminaison locaux suivants :

- Pour une base de données de graphe SPARQL : `https://`*`Neptune-DNS`*`:8182/sparql/stream`.
- Pour une base de données orientée graphe Gremlin ou openCypher : `https://`*`Neptune-DNS`*`:8182/propertygraph/stream` ou `https://`*`Neptune-DNS`*`:8182/pg/stream`.

> **Note**
>
> Le point de terminaison du flux Gremlin (`https://`*`Neptune-DNS`*`:8182/gremlin/stream`) est obsolète, de même que son format de sortie associé (). `GREMLIN_JSON` Il reste pris en charge pour des raisons de rétrocompatibilité, mais il pourrait être supprimé dans les futures versions.

Seule une opération HTTP `GET` est autorisée.

Neptune prend en charge la compression `gzip` de la réponse, à condition que la demande HTTP inclue un en-tête `Accept-Encoding` spécifiant `gzip` comme format de compression accepté (c'est-à-dire, `"Accept-Encoding: gzip"`).

Parameters

- `limit` : long, facultatif. Plage : de 1 à 100 000. Par défaut : 10

  Spécifie le nombre maximal d'enregistrements à renvoyer. Il existe également une limite de taille de 10 Mo pour la réponse qui ne peut pas être modifiée et qui est prioritaire sur le nombre d'enregistrements spécifié dans le paramètre `limit`. La réponse inclut un enregistrement de dépassement de seuil si la limite de 10 Mo a été atteinte.

- `iteratorType` : chaîne, facultatif.

  Ce paramètre peut avoir l'une des valeurs suivantes :

  - `AT_SEQUENCE_NUMBER` (valeur par défaut) : indique que la lecture doit commencer à partir du numéro de séquence d'événement spécifié conjointement par les paramètres `commitNum` et `opNum`.
  - `AFTER_SEQUENCE_NUMBER` : indique que la lecture doit commencer juste après le numéro de séquence d'événement spécifié conjointement par les paramètres `commitNum` et `opNum`.
  - `TRIM_HORIZON` : indique que la lecture doit commencer au niveau du dernier enregistrement non tronqué du système, qui est le plus ancien enregistrement n'ayant pas expiré (pas encore supprimé) dans le flux de journaux des modifications. Ce mode est utile lors du démarrage de l'application, lorsque vous n'avez pas de numéro de séquence d'événement de démarrage spécifique.
  - `LATEST` : indique que la lecture doit commencer au niveau de l'enregistrement le plus récent dans le système, qui est le dernier enregistrement n'ayant pas expiré (pas encore supprimé) dans le flux de journaux des modifications. Cela est utile lorsqu'il est nécessaire de lire les enregistrements à partir du haut actuel des flux afin de ne pas traiter les anciens enregistrements, par exemple lors d'une reprise après sinistre ou d'une mise à niveau sans interruption de service. Notez que dans ce mode, un seul enregistrement est renvoyé au maximum.

- `commitNum` : long, requis lorsque iteratorType est `AT_SEQUENCE_NUMBER` ou `AFTER_SEQUENCE_NUMBER`.

  Numéro de validation de l'enregistrement de départ à lire à partir du flux du journal des modifications.

Ce paramètre est ignoré quand `iteratorType` a la valeur `TRIM_HORIZON` ou `LATEST`.

- `opNum` : long, facultatif (la valeur par défaut est `1`).

  Numéro de séquence d'opération au sein de la validation spécifiée à partir duquel commencer la lecture dans les données du flux du journal des modifications.

Les opérations qui modifient les données de graphe SPARQL génèrent généralement un seul enregistrement de modification par opération. Cependant, les opérations qui modifient les données de graphe Gremlin peuvent générer plusieurs enregistrements de modification par opération, comme dans les exemples suivants :

- `INSERT` : un sommet Gremlin peut avoir plusieurs étiquettes, et un élément Gremlin peut avoir plusieurs propriétés. Un enregistrement de modification distinct est généré pour chaque étiquette et propriété lorsqu'un élément est inséré.
- `UPDATE` : lorsqu'une propriété d'élément Gremlin est modifiée, deux enregistrements de modification sont générés : le premier pour supprimer la valeur précédente et le second pour insérer la nouvelle valeur.
- `DELETE` : un enregistrement de modification distinct est généré pour chaque propriété d'élément supprimée. Par exemple, lorsqu'un arc Gremlin avec des propriétés est supprimé, un enregistrement de modification est généré pour chacune des propriétés, puis un autre enregistrement est généré pour la suppression de l'étiquette d'arc.

  Lorsqu'un sommet Gremlin est supprimé, toutes les propriétés d'arc entrant et sortant sont supprimées en premier, puis viennent les étiquettes d'arc, les propriétés de sommet et enfin les étiquettes de sommet. Chacune de ces suppressions génère un enregistrement de modification.

Format de réponse de l'API Neptune Streams

Une réponse à une demande d'API REST Neptune Streams comporte les champs suivants :

- `lastEventId` : identifiant de séquence de la dernière modification dans la réponse du flux. Un ID d'événement se compose de deux champs : un `commitNum` qui identifie une transaction ayant modifié le graphe et un `opNum` qui identifie une opération spécifique au sein de cette transaction. Voici un exemple :

```
  "eventId": {
```

```
    "commitNum": 12,
    "opNum": 1
  }
```

- `lastTrxTimestamp` : heure à laquelle la validation de la transaction a été demandée, en millisecondes, à partir de l'Unix Epoch.
- `format` : format de sérialisation pour les enregistrements de modification renvoyés. Les valeurs possibles sont `PG_JSON` pour les enregistrements de modification Gremlin ou openCypher et `NQUADS` pour les enregistrements de modification SPARQL.
- `records` : tableau des enregistrements sérialisés du flux de journaux de modifications inclus dans la réponse. Chaque enregistrement du tableau `records` contient les champs suivants :
  - `commitTimestamp` : heure à laquelle la validation de la transaction a été demandée, en millisecondes, à partir de l'Unix Epoch.
  - `eventId` : identifiant de séquence de l'enregistrement de dernière modification du flux.
  - `data`— L'enregistrement sérialisé de Gremlin, SPARQL ou OpenCypher de modification. Les formats de sérialisation de chaque enregistrement sont décrits plus en détail dans la section suivante, Formats de sérialisation dans Neptune Streams.
  - op : opération à l'origine de la modification. Les valeurs valides sont :
    - `ADD`— Indique qu'un nouvel élément (tel qu'un sommet, une arête, une propriété ou un triple) a été ajouté au graphe.
    - `REMOVE`— Indique qu'un élément existant a été supprimé du graphique.

    Neptune exprime une modification apportée à un élément existant sous la forme d'un `REMOVE` enregistrement suivi d'un `ADD` enregistrement.
  - `isLastOp` : présent uniquement si cette opération est la dernière dans sa transaction. Lorsqu'il est présent, il est défini sur `true`. Utile pour s'assurer qu'une transaction est consommée dans son intégralité.
- `totalRecords` : nombre total d'enregistrements dans la réponse.

Par exemple, la réponse suivante renvoie les données de modification Gremlin pour une transaction contenant plusieurs opérations :

```
{
  "lastEventId": {
    "commitNum": 12,
    "opNum": 1
```

```
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1560011610678,
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "ADD"
    }
  ],
  "totalRecords": 1
}
```

La réponse suivante renvoie les données de modification SPARQL pour la dernière opération d'une transaction (opération identifiée par `EventId(97, 1)` dans le numéro de transaction 97).

```
{
  "lastEventId": {
    "commitNum": 97,
    "opNum": 1
  },
  "lastTrxTimestamp": 1561489355102,
  "format": "NQUADS",
  "records": [
    {
      "commitTimestamp": 1561489355102,
      "eventId": {
        "commitNum": 97,
        "opNum": 1
      },
      "data": {
```

```
        "stmt": "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Lire les données du flux Neptune

Les exemples suivants montrent comment lire des enregistrements depuis le point de terminaison du flux du graphe de propriétés. Vous pouvez utiliser AWS CLI le AWS SDK pour Python (Boto3), ouawscurl. curl

AWS CLI

```
aws neptunedata get-propertygraph-stream \
  --endpoint-url https://your-neptune-endpoint:port \
  --iterator-type TRIM_HORIZON \
  --limit 10
```

Pour lire à partir d'une position spécifique dans le flux, utilisez les `--iterator-type` paramètres `--commit-num` et :

```
aws neptunedata get-propertygraph-stream \
  --endpoint-url https://your-neptune-endpoint:port \
  --limit 10 \
  --commit-num 1 \
  --iterator-type AT_SEQUENCE_NUMBER
```

Pour plus d'informations, consultez get-propertygraph-stream dans la référence des commandes. AWS CLI

SDK (Python)

```
import boto3
import json
from botocore.config import Config

client = boto3.client(
    'neptunedata',
```

```
    region_name='us-east-1',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=65)
)

response = client.get_propertygraph_stream(
    limit=10,
    iteratorType='TRIM_HORIZON'
)

print(json.dumps(response['records'], indent=2, default=str))
```

Pour paginer dans le flux, utilisez le `commitNum` et opNum from `lastEventId` dans la réponse :

```
last_event = response['lastEventId']
next_response = client.get_propertygraph_stream(
    limit=10,
    commitNum=last_event['commitNum'],
    opNum=last_event['opNum'],
    iteratorType='AFTER_SEQUENCE_NUMBER'
)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/propertygraph/stream?limit=10 \
  --region us-east-1 \
  --service neptune-db
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Pour plus d'informations sur l'utilisation awscurl de l'authentification IAM, consultezUtiliser `awscurl` avec des informations d'identification temporaires pour se connecter en toute sécurité à un cluster de base de données avec l'authentification IAM activée.

curl

```
curl https://your-neptune-endpoint:port/propertygraph/stream?limit=10
```

> **Note**
>
> L'utilisation curl sans signature SigV4 ne fonctionne que si l'authentification IAM est désactivée sur votre cluster Neptune. Si l'authentification IAM est activée, utilisez awscurl ou à la AWS CLI place.

Pour les graphes SPARQL, remplacez `/propertygraph/stream` par `/sparql/stream` dans les exemples précédents. Lorsque vous utilisez le AWS CLI SDK, utilisez `get-sparql-stream` ou à la `client.get_sparql_stream()` place.

Affichage des données de flux dans un bloc-notes de graphes

Si vous utilisez les blocs-notes Neptune Graph, la magie des `%stream_viewer` lignes fournit une interface visuelle permettant de parcourir les enregistrements de flux de manière interactive sans écrire de code. Exécutez ce qui suit dans une cellule du bloc-notes :

```
%stream_viewer
```

Vous pouvez éventuellement spécifier la langue de requête et le format de page :

```
%stream_viewer sparql --limit 20
```

Pour de plus amples informations, veuillez consulter La magie linéaire `%stream_viewer`.

> **Note**
>
> La `%stream_viewer` magie n'est entièrement prise en charge que sur les versions 1.0.5.1 et antérieures du moteur.

Exceptions de l'API Neptune Streams

Le tableau suivant décrit les exceptions Neptune Streams.

Code d'erreur	Code HTTP	OK pour réessayer ?	Message
`InvalidParameterException`	400	Non	Une valeur non valide ou hors plage a été fournie pour le paramètre d'entrée.
`ExpiredStreamException`	400	Non	Tous les enregistrements demandés dépassent l'âge maximum autorisé et ont expiré.
`ThrottlingException`	500	Oui	La fréquence des demandes dépasse le débit maximum.
`StreamRecordsNotFoundException`	404	Non	La ressource demandée est introuvable. Le flux n'est peut-être pas spécifié correctement.
`MemoryLimitExceededException`	500	Oui	Le traitement de la demande a échoué en raison d'un manque de mémoire, mais pourra être réessayé lorsque le serveur sera moins occupé.

Formats de sérialisation dans Neptune Streams

Amazon Neptune utilise deux formats différents pour sérialiser les données de modification de graphe dans les flux de journaux, selon que le graphe a été créé à l'aide de Gremlin ou de SPARQL.

Les deux formats partagent un format de sérialisation des enregistrements commun, comme décrit dans Format de réponse de l'API Neptune Streams, qui contient les champs suivants :

- `commitTimestamp` : heure à laquelle la validation de la transaction a été demandée, en millisecondes, à partir de l'Unix Epoch.
- `eventId` : identifiant de séquence de l'enregistrement de dernière modification du flux.
- `data`— L'enregistrement sérialisé de Gremlin, SPARQL ou OpenCypher de modification. Les formats de sérialisation de chaque enregistrement sont décrits plus en détail dans les sections suivantes.
- `op` : opération à l'origine de la modification.

Rubriques

- Format de sérialisation des modifications PG_JSON
- Format de sérialisation des modifications SPARQL NQUADS

Format de sérialisation des modifications PG_JSON

> **Note**
>
> Le format de sortie du flux Gremlin (`GREMLIN_JSON`) généré par le point de terminaison du flux Gremlin (`https://Neptune-DNS:8182/gremlin/stream`) est obsolète. Il est remplacé par PG_JSON, qui est actuellement identique à `GREMLIN_JSON`.

Un enregistrement de modification Gremlin ou openCypher, contenu dans le champ `data` d'une réponse de flux de journaux, comporte les champs suivants :

- `id` : chaîne, obligatoire.

  ID de l'élément Gremlin ou openCypher.

- `type` : chaîne, obligatoire.

  Type de cet élément Gremlin ou openCypher. Doit être l'un des suivants :

  - `vl` : étiquette du sommet pour Gremlin ; étiquette du nœud pour openCypher.
  - `vp` : propriétés du sommet pour Gremlin ; propriétés du nœud pour openCypher.

  - `e` : arête et étiquette d'arête pour Gremlin ; relation et type de relation pour openCypher.
  - `ep` : propriétés de l'arête pour Gremlin ; propriétés de la relation pour openCypher.
- `key` : chaîne, obligatoire.

  Nom de la propriété. Pour les étiquettes d'élément, il s'agit de « label ».
- `value` : objet `value`, obligatoire.

  Il s'agit d'un objet JSON qui contient un champ `value` pour la valeur elle-même et un champ `datatype` pour le type de données JSON de cette valeur.

```
  "value": {
    "value": "the new value",
    "dataType": "the JSON datatype of the new value"
  }
```

- `from` : chaîne, facultatif.

  S'il s'agit d'une arête (type="e"), ID du sommet source ou du nœud source correspondant.
- `to` : chaîne, facultatif.

  S'il s'agit d'une arête (type="e"), ID du sommet cible ou du nœud cible correspondant.

Exemples Gremlin

- Voici un exemple d'étiquette de sommet Gremlin.

```
{
  "id": "an ID string",
  "type": "vl",
  "key": "label",
  "value": {
    "value": "the new value of the vertex label",
    "dataType": "String"
  }
}
```

- Voici un exemple de propriété de sommet Gremlin.

```
{
  "id": "an ID string",
```

```
  "type": "vp",
  "key": "the property name",
  "value": {
    "value": "the new value of the vertex property",
    "dataType": "the datatype of the vertex property"
  }
}
```

- Voici un exemple d'arc Gremlin.

```
{
  "id": "an ID string",
  "type": "e",
  "key": "label",
  "value": {
    "value": "the new value of the edge",
    "dataType": "String"
  },
  "from": "the ID of the corresponding "from" vertex",
  "to": "the ID of the corresponding "to" vertex"
}
```

Exemples openCypher

- Voici un exemple d'étiquette de nœud openCypher.

```
{
  "id": "an ID string",
  "type": "vl",
  "key": "label",
  "value": {
    "value": "the new value of the node label",
    "dataType": "String"
  }
}
```

- Voici un exemple de propriété de nœud openCypher.

```
{
  "id": "an ID string",
  "type": "vp",
  "key": "the property name",
```

```
  "value": {
    "value": "the new value of the node property",
    "dataType": "the datatype of the node property"
  }
}
```

- Voici un exemple de relation openCypher.

```
{
  "id": "an ID string",
  "type": "e",
  "key": "label",
  "value": {
    "value": "the new value of the relationship",
    "dataType": "String"
  },
  "from": "the ID of the corresponding source node",
  "to": "the ID of the corresponding target node"
}
```

Format de sérialisation des modifications SPARQL NQUADS

Neptune enregistre les modifications apportées aux quads SPARQL dans le graphe en utilisant le N-QUADS langage RDF (Resource Description Framework) défini dans la spécification W3C RDF 1.1. N-Quads

Le `data` champ de l'enregistrement des modifications contient simplement un `stmt` champ contenant une N-QUADS instruction exprimant le quadruple modifié, comme dans l'exemple suivant.

```
"stmt" : "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
```

Exemples Neptune Streams

Les exemples suivants montrent comment accéder aux données de flux des journaux de modifications dans Amazon Neptune.

Rubriques

- Journal des modifications AT_SEQUENCE_NUMBER
- Journal des modifications AFTER_SEQUENCE_NUMBER

- Journal des modifications TRIM_HORIZON
- DERNIER journal des modifications
- Journal des modifications Compression

Journal des modifications AT_SEQUENCE_NUMBER

L'exemple suivant montre un journal des modifications Gremlin ou openCypher `AT_SEQUENCE_NUMBER`.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AT_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
    {
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "commitTimestamp": 1560011610678,
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Celui-ci montre un exemple SPARQL de journal des modifications `AT_SEQUENCE_NUMBER`.

```
curl -s "https://localhost:8182/sparql/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AT_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1571252030566,
  "format": "NQUADS",
  "records": [
    {
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "commitTimestamp": 1571252030566,
      "data": {
        "stmt": "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Journal des **modifications AFTER_SEQUENCE_NUMBER**

L'exemple suivant montre un journal des modifications Gremlin ou openCypher `AFTER_SEQUENCE_NUMBER`.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AFTER_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
    "commitNum": 2,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011633768,
  "format": "PG_JSON",
```

```
  "records": [
    {
      "commitTimestamp": 1560011633768,
      "eventId": {
        "commitNum": 2,
        "opNum": 1
      },
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "REMOVE",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Journal des **modifications TRIM_HORIZON**

L'exemple suivant montre un journal des modifications Gremlin ou openCypher `TRIM_HORIZON`.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&iteratorType=TRIM_HORIZON" |jq
{
  "lastEventId": {
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1560011610678,
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
```

```
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

DERNIER journal des modifications

L'exemple suivant montre un journal des modifications Gremlin ou openCypher LATEST. Notez que les paramètres d'API `limit`, `commitNum` et opNum sont totalement facultatifs.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?iteratorType=LATEST" | jq
{
  "lastEventId": {
    "commitNum": 21,
    "opNum": 4
  },
  "lastTrxTimestamp": 1634710497743,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1634710497743,
      "eventId": {
        "commitNum": 21,
        "opNum": 4
      },
      "data": {
        "id": "24be4e2b-53b9-b195-56ba-3f48fa2b60ac",
        "type": "e",
        "key": "label",
        "value": {
          "value": "created",
          "dataType": "String"
```

```
        },
        "from": "4",
        "to": "5"
      },
      "op": "REMOVE",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Journal des modifications Compression

L'exemple suivant montre un journal des modifications de compression Gremlin ou openCypher.

```
curl -sH \
  "Accept-Encoding: gzip" \
  "https://Neptune-DNS:8182/propertygraph/stream?limit=1&commitNum=1" \
  -H "Accept-Encoding: gzip" \
  -v |gunzip -|jq
> GET /propertygraph/stream?limit=1 HTTP/1.1
> Host: localhost:8182
> User-Agent: curl/7.64.0
> Accept: /
> Accept-Encoding: gzip
*> Accept-Encoding: gzip*
>
< HTTP/1.1 200 OK
< Content-Type: application/json; charset=UTF-8
< Connection: keep-alive
*< content-encoding: gzip*
< content-length: 191
<
{ [191 bytes data]
Connection #0 to host localhost left intact
{
  "lastEventId": "1:1",
  "lastTrxTimestamp": 1558942160603,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1558942160603,
      "eventId": "1:1",
```

```
      "data": {
        "id": "v1",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "person",
          "dataType": "String"
        }
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Utilisation AWS CloudFormation pour configurer la Neptune-to-Neptune réplication avec l'application Streams Consumer

Vous pouvez utiliser un CloudFormation modèle pour configurer l'application grand public Neptune Streams afin de prendre en charge Neptune-to-Neptune la réplication.

Rubriques

- Choisissez un CloudFormation modèle pour votre région
- Ajout de détails sur la pile de consommateurs de flux Neptune que vous créez
- Exécutez le CloudFormation Modèle
- Pour mettre à jour l'interrogateur de flux avec les derniers artefacts Lambda

Choisissez un CloudFormation modèle pour votre région

Pour lancer la CloudFormation pile appropriée sur la CloudFormation console, cliquez sur l'un des boutons Launch Stack du tableau suivant, en fonction de la AWS région que vous souhaitez utiliser.

Région	Afficher	Lancer
USA Est (Virginie du Nord)	Afficher	Launch Stack

Région	Afficher	Lancer
USA Est (Ohio)	Afficher	
USA Ouest (Californie du Nord)	Afficher	
USA Ouest (Oregon)	Afficher	
Canada (Centre)	Afficher	
Amérique du Sud (São Paulo)	Afficher	
Europe (Stockholm)	Afficher	
Europe (Irlande)	Afficher	
Europe (Londres)	Afficher	
Europe (Paris)	Afficher	
Europe (Francfort)	Afficher	
Middle East (Bahrain)	Afficher	
Moyen-Orient (EAU)	Afficher	
Israël (Tel Aviv)	Afficher	
Afrique (Le Cap)	Afficher	
Asie-Pacifique (Tokyo)	Afficher	

Région	Afficher	Lancer
Asie-Pacifique (Hong Kong)	Afficher	
Asie-Pacifique (Séoul)	Afficher	
Asie-Pacifique (Singapour)	Afficher	
Asie-Pacifique (Sydney)	Afficher	
Asie-Pacifique (Mumbai)	Afficher	
Chine (Pékin)	Afficher	
Chine (Ningxia)	Afficher	
AWS GovCloud (US-West)	Afficher	
AWS GovCloud (US-East)	Afficher	

Sur la page Créer une pile choisissez Suivant.

Ajout de détails sur la pile de consommateurs de flux Neptune que vous créez

La page Spécifier les détails de la pile fournit les propriétés et les paramètres que vous pouvez utiliser pour contrôler la configuration de l'application :

Nom de la pile : nom de la nouvelle CloudFormation pile que vous créez. Vous pouvez généralement utiliser la valeur par défaut, `NeptuneStreamPoller`.

Sous Paramètres, indiquez ce qui suit :

Configuration réseau du VPC sur lequel le consommateur de flux est exécuté

- **`VPC`** : indiquez le nom du VPC où la fonction Lambda d'interrogation sera exécutée.
- **`SubnetIDs`** : sous-réseaux avec lesquels une interface réseau est établie. Ajoutez des sous-réseaux correspondant à votre cluster Neptune.
- **`SecurityGroupIds`** : fournissez les ID des groupes de sécurité qui accordent un accès entrant en écriture à votre cluster de bases de données Neptune source.
- **`RouteTableIds`** : nécessaire pour créer un point de terminaison Amazon DynamoDB dans votre VPC Neptune, si vous n'en avez pas déjà un. Vous devez fournir une liste séparée par des virgules des ID de table de routage associés aux sous-réseaux.
- **`CreateDDBVPCEndPoint`** : valeur booléenne qui est `true` par défaut et qui indique s'il est nécessaire ou non de créer un point de terminaison de VPC Dynamo DB. Vous n'avez besoin de la remplacer par `false` que si vous avez déjà créé un point de terminaison DynamoDB dans votre VPC.
- **`CreateMonitoringEndPoint`** : valeur booléenne qui est `true` par défaut et qui indique s'il est nécessaire ou non de créer un point de terminaison de VPC de surveillance. Vous n'avez besoin de la modifier en `false` que si vous avez déjà créé un point de terminaison de surveillance dans votre VPC.

Interrogateur de flux

- **`ApplicationName`** : vous pouvez généralement conserver la valeur par défaut (`NeptuneStream`). Si vous utilisez un nom différent, il doit être unique.
- **`LambdaMemorySize`** : utilisé pour définir la taille de mémoire disponible pour la fonction d'interrogateur Lambda. La valeur par défaut est 2 048 mégaoctets.
- **`LambdaRuntime`** : langage utilisé dans la fonction Lambda qui extrait les éléments du flux Neptune. Vous pouvez définir ce paramètre sur `python3.9` ou sur `java8`.
- **`LambdaS3Bucket`** : compartiment Amazon S3 qui contient les artefacts de code Lambda. Laissez cette case vide, sauf si vous utilisez une fonction d'interrogation Lambda personnalisée qui se charge à partir d'un compartiment Amazon S3 différent.
- **`LambdaS3Key`** : clé Amazon S3 qui correspond aux artefacts de code Lambda. Laissez ce champ vide, sauf si vous utilisez une fonction d'interrogation Lambda personnalisée.
- **`LambdaLoggingLevel`** : en général, conservez la valeur par défaut, qui est `INFO`.

- **`ManagedPolicies`** : répertorie les politiques gérées à utiliser pour exécuter la fonction Lambda. En général, laissez ce champ vide, sauf si vous utilisez une fonction d'interrogation Lambda personnalisée.
- **`StreamRecordsHandler`** : en général, laissez ce champ vide, sauf si vous utilisez un gestionnaire personnalisé pour les enregistrements dans les flux Neptune.
- **`StreamRecordsBatchSize`** : nombre maximal d'enregistrements à extraire du flux. Vous pouvez utiliser ce paramètre pour régler les performances. La valeur par défaut (`5000`) est un bon endroit pour commencer. Le maximum autorisé est de 10 000. Plus le nombre est élevé, moins les appels réseau sont nécessaires pour lire les enregistrements du flux, mais plus la mémoire est nécessaire pour traiter les enregistrements. Des valeurs plus faibles pour ce paramètre se traduisent par une baisse du débit.
- **`MaxPollingWaitTime`** : temps d'attente maximal entre deux interrogations (en secondes). Détermine la fréquence à laquelle l'interrogateur Lambda est appelé pour interroger les flux Neptune. Définissez cette valeur sur 0 pour l'interrogation continue. La valeur maximale est de 3 600 secondes (1 heure). La valeur par défaut (60 secondes) est un bon point de départ, en fonction de la vitesse à laquelle vos données de graphe changent.
- **`MaxPollingInterval`** : période d'interrogation continue maximale (en secondes). Utilisez ce paramètre afin de définir un délai d'expiration pour la fonction d'interrogation Lambda. Cette valeur doit être comprise entre 5 et 900 secondes. La valeur par défaut (600 secondes) est un bon point de départ.
- **`StepFunctionFallbackPeriod`**— Le nombre d'unités de période de transition entre fonction et période de repli devant attendre le sondeur, après quoi la fonction d'étape est appelée via Amazon CloudWatch Events pour se remettre d'une défaillance. La valeur par défaut (5 minutes) est un bon point de départ.
- **`StepFunctionFallbackPeriodUnit`** : unités de temps utilisées pour mesurer l'élément `StepFunctionFallbackPeriodUnit` précédent (`minutes`, `hours` ou `days`). La valeur par défaut (`minutes`) est généralement suffisante.
- **`StartingCheckpoint`**— Le point de contrôle de départ du stream poller. La valeur par défaut est `0:0`, ce qui signifie que l'interrogateur de flux doit commencer au début du flux Neptune.
- **`StreamPollerInitialState`**— L'état initial du sondeur. La valeur par défaut est `ENABLED`, ce qui signifie que la réplication du flux débutera dès que la création complète de la pile sera terminée.

Flux Neptune

- **`NeptuneStreamEndpoint`** : (obligatoire) point de terminaison du flux source Neptune. Ce paramètre prend l'une des deux formes suivantes :
  - **`https://`*`your DB cluster`*`:`*`port`*`/propertygraph/stream`** (ou son alias, `https://`*`your DB cluster`*`:`*`port`*`/pg/stream`).
  - **`https://`*`your DB cluster`*`:`*`port`*`/sparql/stream`**.
- **`Neptune Query Engine`** : choisissez Gremlin, openCypher ou SPARQL.
- **`IAMAuthEnabledOnSourceStream`** : si votre cluster de bases de données Neptune utilise l'authentification IAM, définissez ce paramètre sur `true`.
- **`StreamDBClusterResourceId`** : si votre cluster de bases de données Neptune utilise l'authentification IAM, définissez ce paramètre sur l'ID de ressource du cluster. L'ID de ressource n'est pas le même que l'ID de cluster. Elle prend plutôt la forme : `cluster-` suivi de 28 caractères alphanumériques. Vous pouvez le trouver sous Détails du cluster dans la console Neptune.

Cluster de bases de données Neptune cible

- **`TargetNeptuneClusterEndpoint`** : point de terminaison (nom d'hôte uniquement) du cluster de sauvegarde cible.

  Notez que si vous spécifiez `TargetNeptuneClusterEndpoint`, vous ne pouvez pas renseigner aussi `TargetSPARQLUpdateEndpoint`.
- **`TargetNeptuneClusterPort`** : numéro de port du cluster cible.

  Notez que si vous spécifiez `TargetSPARQLUpdateEndpoint`, le paramètre correspondant à `TargetNeptuneClusterPort` est ignoré.
- **`IAMAuthEnabledOnTargetCluster`** : défini sur true si l'authentification IAM doit être activée sur le cluster cible.
- **`TargetAWSRegion`**— La AWS région du cluster de sauvegarde cible, telle que`us-east-1`). Vous devez fournir ce paramètre uniquement lorsque la AWS région du cluster de sauvegarde cible est différente de celle du cluster source Neptune, comme dans le cas d'une réplication entre régions. Si les régions source et cible sont identiques, ce paramètre est facultatif.

  Notez que si la `TargetAWSRegion` valeur n'est pas une AWS région valide prise en charge par Neptune, le processus échoue.

- **`TargetNeptuneDBClusterResourceId`** : facultatif : nécessaire uniquement lorsque l'authentification IAM est activée sur le cluster de bases de données cible. Défini sur l'ID de ressource du cluster cible.
- **`SPARQLTripleOnlyMode`** : indicateur booléen qui détermine si le mode triple uniquement est activé. En mode triple uniquement, il n'y a pas de réplication de graphes nommés. La valeur par défaut est `false`.
- **`TargetSPARQLUpdateEndpoint`** : URL du point de terminaison cible pour la mise à jour SPARQL, telle que `https://abc.com/xyz`. Ce point de terminaison peut être n'importe quel magasin SPARQL qui prend en charge les quadruplets ou les triplets.

  Notez que si vous spécifiez `TargetSPARQLUpdateEndpoint`, vous ne pouvez pas renseigner aussi `TargetNeptuneClusterEndpoint`, et le paramètre `TargetNeptuneClusterPort` est ignoré.
- **`BlockSparqlReplicationOnBlankNode`** — Indicateur booléen qui, s'il est défini sur`true`, arrête la réplication des données BlankNode dans SPARQL (RDF). La valeur par défaut est `false`.

alerte

- **`Required to create Cloud watch Alarm`**— Réglez ce `true` paramètre sur si vous souhaitez créer une CloudWatch alarme pour la nouvelle pile.
- **`SNS Topic ARN for Cloudwatch Alarm Notifications`**— L'ARN de la rubrique SNS où les notifications CloudWatch d'alarme doivent être envoyées (uniquement nécessaire si les alarmes sont activées).
- **`Email for Alarm Notifications`** : adresse e-mail à laquelle les notifications d'alarme doivent être envoyées (uniquement si les alarmes sont activées).

Pour la destination de la notification d'alarme, vous pouvez ajouter la rubrique SNS uniquement, l'adresse e-mail uniquement ou les deux.

Exécutez le CloudFormation Modèle

Vous pouvez maintenant terminer le processus de provisionnement d'une instance d'application grand public de flux Neptune comme suit :

1. Dans CloudFormation, sur la page Spécifier les détails de la pile, choisissez Next.

2. Dans la page Options, choisissez Suivant.
3. Sur la page Vérification, cochez la première case pour accepter que CloudFormation créera des ressources IAM. Cochez la deuxième case pour confirmer `CAPABILITY_AUTO_EXPAND` comme nouvelle pile.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` accepte explicitement que les macros soient étendues lors de la création de la pile, sans révision préalable. Les utilisateurs créent souvent un jeu de modifications à partir d'un modèle traité, de sorte que les modifications apportées par les macros puissent être révisées avant la création de la pile. Pour plus d'informations, consultez l' CloudFormation CreateStackAPI dans le Guide de référence des AWS CloudFormation API.

   Ensuite, choisissez Créer.

Pour mettre à jour l'interrogateur de flux avec les derniers artefacts Lambda

Pour mettre à jour l'interrogateur de flux avec les derniers artefacts Lambda, procédez comme suit :

1. Dans le AWS Management Console, naviguez jusqu'à la CloudFormation pile parent principale CloudFormation et sélectionnez-la.
2. Sélectionnez l'option Mettre à jour correspondant à la pile.
3. Sélectionnez Remplacer le modèle actuel.
4. Pour la source du modèle, choisissez URL Amazon S3 et indiquez l'URL S3 suivante :

   ```
   https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
   neptune_to_neptune.json
   ```

5. Sélectionnez Next sans modifier aucun CloudFormation paramètre.
6. Choisissez Mettre à jour la pile.

La pile met à jour les artefacts Lambda avec les artefacts les plus récents.

Utilisation de la réplication des flux Neptune entre régions pour la reprise après sinistre

Neptune propose deux méthodes pour implémenter des fonctionnalités de basculement entre régions :

- Cross-region copie instantanée et restauration
- Utilisation des flux Neptune pour répliquer les données entre deux clusters situés dans deux régions différentes.

Cross-region la copie et la restauration d'instantanés présentent les coûts opérationnels les plus faibles pour la restauration d'un cluster Neptune dans une autre région. Cependant, la copie d'un instantané entre régions peut nécessiter un temps de transfert de données important, car un instantané implique une sauvegarde complète du cluster Neptune. Par conséquent, la copie et la restauration d'instantanés entre régions peuvent être utilisées dans des scénarios qui nécessitent uniquement un objectif de point de reprise (RPO) de plusieurs heures et un objectif de délai de reprise (RTO) de plusieurs heures.

Un objectif de point de reprise (RPO) est mesuré en fonction du temps écoulé entre les sauvegardes. Il définit la quantité de données qui peut être perdue entre le moment où la dernière sauvegarde a été effectuée et le moment où la base de données est restaurée.

Un objectif de délai de reprise (RTO) est mesuré en fonction du temps nécessaire pour effectuer une opération de restauration. C'est le temps qu'il faut au cluster de bases de données pour basculer vers une base de données restaurée après une panne.

Neptune Streams permet de synchroniser à tout moment un cluster Neptune de sauvegarde avec le cluster de production principal. En cas de panne, la base de données bascule vers le cluster de sauvegarde. Cela réduit le RPO et le RTO à quelques minutes, car les données sont constamment copiées sur le cluster de sauvegarde, qui est immédiatement disponible en tant que cible de basculement à tout moment.

L'inconvénient lié à l'utilisation des flux Neptune de cette manière est que la charge opérationnelle requise pour gérer les composants de réplication et le coût associé à la mise en ligne permanente d'un deuxième cluster de bases de données Neptune peuvent être significatifs.

Configuration de la Neptune-to-Neptune réplication

Votre cluster de bases de données de production principal se trouve dans un VPC dans une région source donnée. Trois éléments principaux doivent être répliqués ou émulés dans une autre région de reprise aux fins de la reprise après sinistre :

- Les données stockées dans le cluster.
- La configuration du cluster principal. Authentification IAM ou non, chiffrement ou non, paramètres de cluster de bases de données, paramètres d'instance, tailles d'instance, etc..
- La topologie réseau qu'il utilise, y compris le VPC cible, ses groupes de sécurité, etc.

Vous pouvez utiliser les API de gestion Neptune telles que les suivantes pour recueillir ces informations :

- `DescribeDBClusters`
- `DescribeDBInstances`
- `DescribeDBClusterParameters`
- `DescribeDBParameters`
- `DescribeVpcs`

Avec les informations que vous collectez, vous pouvez utiliser la procédure suivante pour configurer un cluster de sauvegarde dans une autre région, vers lequel votre cluster de production pourra basculer en cas de panne.

Activer les flux Neptune

Vous pouvez utiliser Modifier DBCluster ParameterGroup pour définir le paramètre `neptune_streams` sur 1. Redémarrez ensuite toutes les instances du cluster de bases de données pour que les modifications prennent effet.

Il est conseillé d'effectuer au moins une opération d'ajout ou de mise à jour au niveau du cluster de bases de données source une fois que Neptune Streams a été activé. Cela fournit au flux de modifications des points de données qui pourront être référencés ultérieurement lors de la resynchronisation du cluster de production avec le cluster de sauvegarde.

Créez un nouveau VPC dans la région où vous souhaitez configurer votre cluster de sauvegarde

Avant de créer un cluster de bases de données Neptune dans une région différente de celle du cluster principal, vous devez établir un nouveau VPC dans la région cible afin d'y héberger ce cluster. La connectivité entre le cluster principal et le cluster de sauvegarde est établie par le biais de l'appairage de VPC, qui utilise le trafic entre les sous-réseaux privés de différents VPC. Toutefois, pour établir l'appairage entre deux VPC, ceux-ci ne doivent pas avoir de blocs d'adresse CIDR ni d'espaces d'adresses IP qui se chevauchent. Autrement dit, vous ne pouvez pas simplement utiliser le VPC par défaut dans les deux régions, car le bloc d'adresse CIDR d'un VPC par défaut est toujours le même (`172.31.0.0/16`).

Vous pouvez utiliser un VPC existant dans la région cible à condition qu'il respecte les conditions suivantes :

- Il ne dispose pas d'un bloc d'adresse CIDR qui chevauche celui du VPC où se trouve le cluster principal.
- Il n'a pas déjà été appairé avec un autre VPC possédant le même bloc d'adresse CIDR que le VPC où se trouve le cluster principal.

Si aucun VPC adapté n'est disponible dans la région cible, créez-en un à l'aide de l'API Amazon EC2 `CreateVpc`.

Créez un instantané de votre cluster principal et restaurez-le dans la région de sauvegarde cible

À présent, créez un cluster Neptune, qui est une copie du cluster de production, dans un VPC approprié dans la région de sauvegarde cible, :

Créez une copie du cluster de production dans la région de sauvegarde

1. Dans la région de sauvegarde cible, recréez les paramètres et les groupes de paramètres utilisés par votre cluster de bases de données de production. Pour ce faire, utilisez `CreateDBClusterParameterGroup`, `CreateDBParameterGroup`, `ModifyDBClusterParameterGroup` et `ModifyDBParameterGroup`.

   Notez que les API `CopyDBParameterGroup` et `CopyDBClusterParameterGroup` ne prennent actuellement pas en charge la copie entre régions.

2. Utilisez `CreateDBClusterSnapshot` pour créer un instantané du cluster de production dans le VPC de la région de production.
3. Utilisez `CopyDBClusterSnapshot` pour copier l'instantané sur le VPC de la région de sauvegarde cible.
4. Utilisez `RestoreDBClusterFromSnapshot` pour créer un cluster de bases de données dans le VPC de la région de sauvegarde cible à l'aide de l'instantané copié. Utilisez les paramètres de configuration et les paramètres que vous avez copiés depuis le cluster de production principal.
5. Le nouveau cluster Neptune existe désormais, mais ne contient aucune instance. `CreateDBInstance`À utiliser pour créer une nouvelle primary/writer instance ayant le même type et la même taille que l'instance Writer de votre cluster de production. Il n'est pas nécessaire de créer des répliques de lecture supplémentaires à ce stade, à moins que votre instance de sauvegarde ne soit utilisée pour effectuer la lecture I/O dans la région cible avant un basculement.

Établissez un peering VPC entre le VPC de votre cluster principal et le VPC de votre nouveau cluster de sauvegarde

En configurant l'appairage de VPC, vous permettez au VPC du cluster principal de communiquer avec le VPC du cluster de sauvegarde comme s'il s'agissait d'un réseau privé unique. Pour cela, effectuez les opérations suivantes :

1. À partir du VPC du cluster de production, appelez l'API `CreateVpcPeeringConnection` pour établir la connexion d'appairage.
2. À partir du VPC du cluster de sauvegarde cible, appelez l'API `AcceptVpcPeeringConnection` pour accepter la connexion d'appairage.
3. À partir du VPC du cluster de production, utilisez l'API `CreateRoute` pour ajouter une route à la table de routage du VPC. Cette dernière redirigera tout le trafic vers le bloc d'adresse CIDR du VPC cible afin qu'il utilise la liste des préfixes d'appairage de VPC.
4. De même, à partir du VPC du cluster de sauvegarde cible, utilisez l'API `CreateRoute` pour ajouter une route à la table de routage du VPC. Cette dernière acheminera le trafic vers le VPC du cluster principal.

Configuration de l'infrastructure de réplication des flux Neptune

Maintenant que les deux clusters sont déployés et que la communication réseau entre les deux régions est établie, utilisez le Neptune-to-Neptune CloudFormation modèle pour déployer la fonction Lambda de Neptune Streams Consumer avec l'infrastructure supplémentaire qui prend en charge la réplication des données. Procédez dans le VPC du cluster de production principal.

Les paramètres que vous devrez fournir pour cette CloudFormation pile sont les suivants :

- **`NeptuneStreamEndpoint`** : point de terminaison du flux pour le cluster principal, au format URL. Par exemple : `https://(cluster name):8182/pg/stream`.
- **`QueryEngine`** : les valeurs possibles sont `gremlin`, `sparql` ou `openCypher`.
- **`RouteTableIds`** : vous permet d'ajouter des routes à la fois pour un point de terminaison de VPC DynamoDB et pour un point de terminaison de VPC de surveillance.

  Deux paramètres supplémentaires, à savoir `CreateMonitoringEndpoint` et `CreateDynamoDBEndpoint`, doivent également être définis sur true s'ils n'existent pas déjà sur le VPC du cluster principal. S'ils existent déjà, assurez-vous qu'ils sont définis sur false, sinon la CloudFormation création échouera.
- **`SecurityGroupIds`** : spécifie le groupe de sécurité utilisé par le consommateur Lambda pour communiquer avec le point de terminaison de flux Neptune du cluster principal.

  Dans le cluster de sauvegarde cible, attachez un groupe de sécurité qui autorise le trafic provenant de ce groupe de sécurité.
- **`SubnetIds`** : liste d'ID de sous-réseau dans le VPC du cluster principal, qui peut être utilisée par le consommateur Lambda pour communiquer avec le cluster principal.
- **`TargetNeptuneClusterEndpoint`** : point de terminaison (nom d'hôte uniquement) du cluster de sauvegarde cible.
- **`TargetAWSRegion`**— La AWS région du cluster de sauvegarde cible, telle que`us-east-1`). Vous devez fournir ce paramètre uniquement lorsque la AWS région du cluster de sauvegarde cible est différente de celle du cluster source Neptune, comme dans le cas d'une réplication entre régions. Si les régions source et cible sont identiques, ce paramètre est facultatif.

  Notez que si la `TargetAWSRegion` valeur n'est pas une AWS région valide prise en charge par Neptune, le processus échoue.
- **`VPC`** : ID du VPC du cluster principal.

Tous les autres paramètres peuvent conserver leurs valeurs par défaut.

Une fois le CloudFormation modèle déployé, Neptune commencera à répliquer toutes les modifications du cluster principal vers le cluster de sauvegarde. Vous pouvez surveiller cette réplication dans les CloudWatch journaux générés par la fonction Lambda Consumer.

Autres considérations

- Si vous devez utiliser l'authentification IAM entre le cluster principal et le cluster de sauvegarde, vous pouvez également la configurer lorsque vous appelez le modèle. CloudFormation
- Si le chiffrement au repos est activé sur le cluster principal, réfléchissez à la manière de gérer les clés KMS associées lors de la copie de l'instantané dans la région cible, ainsi que sur la manière d'associer une nouvelle clé KMS dans la région cible.
- Une bonne pratique consiste à utiliser des DNS CNAME devant les points de terminaison Neptune utilisés dans vos applications. Ensuite, si vous devez basculer manuellement vers le cluster de sauvegarde cible, ces CNAME peuvent être modifiés pour pointer vers les points de terminaison de l' and/or instance de cluster cible.

Recherche en texte intégral dans Amazon Neptune à l'aide d'Amazon Service OpenSearch

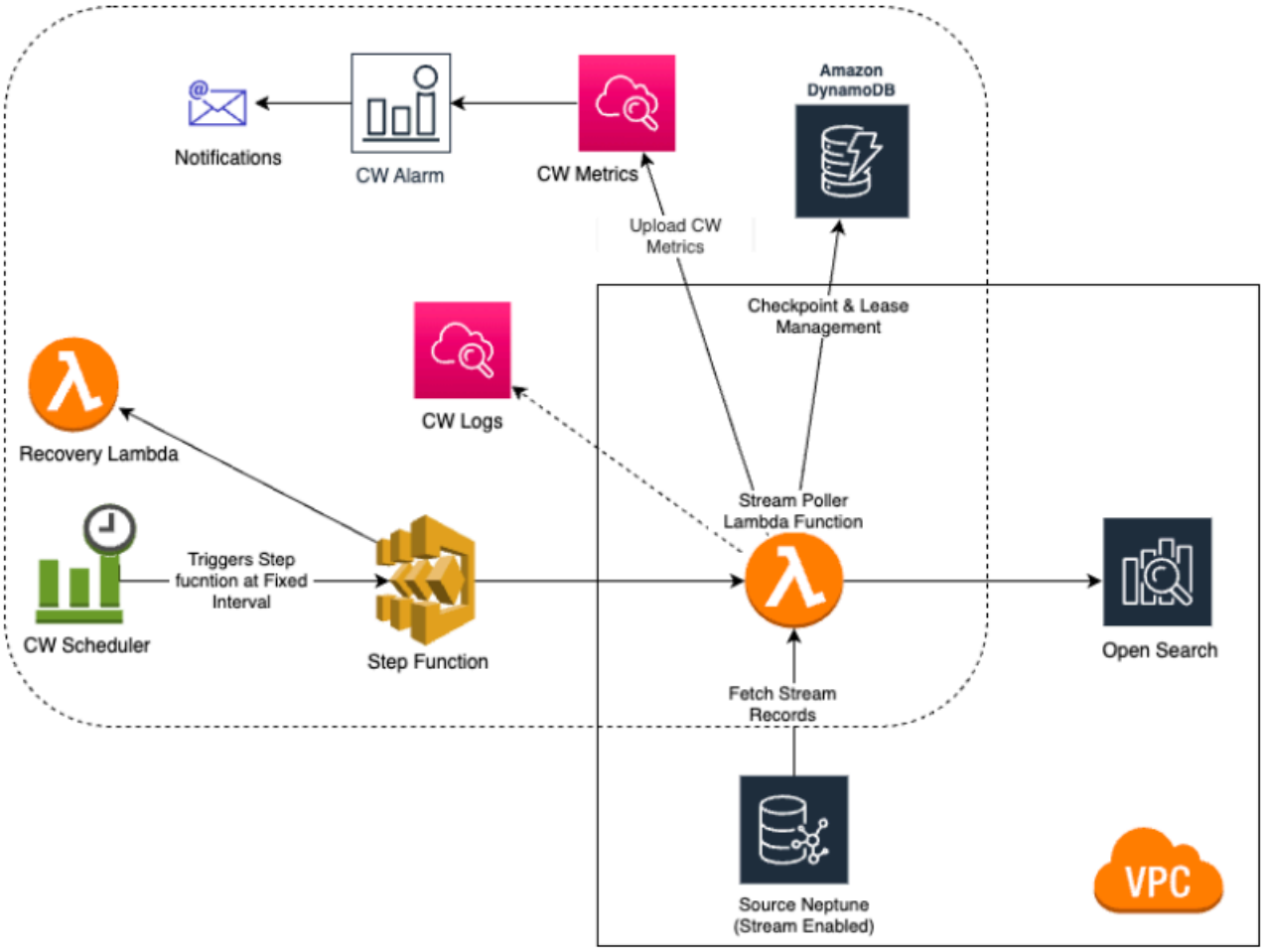
Neptune s'intègre à Amazon OpenSearch Service (OpenSearch Service) pour prendre en charge la recherche en texte intégral dans les requêtes Gremlin et SPARQL.

À partir de la version 1.3.0.0 du moteur, Amazon Neptune prend en charge l'utilisation d' OpenSearch Amazon Service Serverless pour la recherche en texte intégral dans les requêtes Gremlin et SPARQL.

> **Note**
>
> Lors de l'intégration à Amazon OpenSearch Service, Neptune nécessite la version 7.1 ou supérieure d'Elasticsearch et fonctionne avec les versions OpenSearch 2.3, 2.5 et supérieures. Neptune fonctionne également avec OpenSearch Serverless.

Vous pouvez utiliser Neptune avec un cluster de OpenSearch services existant qui a été renseigné conformément au. Modèle de données Neptune pour les données OpenSearch Vous pouvez également créer un domaine de OpenSearch service lié à Neptune à l'aide d'une AWS CloudFormation pile.

> **Important**
>
> Le processus de OpenSearch réplication Neptune décrit ici ne réplique pas les nœuds vides. Il s'agit d'une limite importante à noter.
> En outre, si vous activez le contrôle d'accès détaillé sur votre OpenSearch cluster, vous devez également activer l'authentification IAM dans votre base de données Neptune.



Rubriques

- Neptune-to-OpenSearch Réplication Amazon
- Réplication vers le mode OpenSearch sans serveur
- Interrogation depuis un OpenSearch cluster sur lequel le contrôle d'accès détaillé (FGAC) est activé
- Utilisation de la syntaxe de requête Apache Lucene dans les requêtes de recherche en texte intégral Neptune
- Modèle de données Neptune pour les données OpenSearch
- Paramètres de recherche en texte intégral Neptune
- OpenSearch Indexation sans chaîne dans Amazon Neptune
- Full-text-search exécution de requêtes dans Amazon Neptune
- Exemples de requêtes SPARQL utilisant la recherche en texte intégral dans Neptune
- Utilisation de la recherche en texte intégral Neptune dans des requêtes Gremlin

- Résolution des problèmes liés à la recherche Neptune en texte intégral

Neptune-to-OpenSearch Réplication Amazon

Amazon Neptune prend en charge la recherche en texte intégral dans les requêtes Gremlin et SPARQL à l'aide d' OpenSearch Amazon Service (Service). OpenSearch Vous pouvez utiliser une AWS CloudFormation pile pour lier un domaine OpenSearch de service à Neptune. Le CloudFormation modèle crée une instance d'application streams-consumer qui assure Neptune-to-OpenSearch la réplication.

Avant de commencer, vous avez besoin d'un cluster de base de données Neptune existant sur lequel les flux sont activés pour servir de source, et d'un domaine de OpenSearch service pour servir de cible de réplication.

Si vous disposez déjà d'un domaine de OpenSearch service cible accessible par Lambda dans le VPC où se trouve votre cluster de base de données Neptune, le modèle peut utiliser celui-ci. Sinon, vous devez en créer un nouveau.

> **Note**
>
> Le OpenSearch cluster et la fonction Lambda que vous créez doivent être situés dans le même VPC que votre cluster de base de données Neptune, et le cluster OpenSearch doit être configuré en mode VPC (et non en mode Internet).

Nous vous recommandons d'utiliser une instance Neptune nouvellement créée pour l'utiliser avec OpenSearch le Service. Si vous utilisez une instance existante qui contient déjà des données, vous devez effectuer une synchronisation des données de OpenSearch service avant d'effectuer des requêtes, faute de quoi il peut y avoir des incohérences dans les données. Ce GitHub projet fournit un exemple de la manière d'effectuer la synchronisation : Exporter Neptune vers OpenSearch ()https://github.com/awslabs/amazon-neptune-tools/tree/master/export-neptune-to-elasticsearch.

> **Important**
>
> Lors de l'intégration à Amazon OpenSearch Service, Neptune nécessite la version 7.1 ou supérieure d'Elasticsearch et fonctionne avec les versions OpenSearch 2.3, 2.5 et les futures versions compatibles d'Opensearch.

> **Note**
>
> À partir de la version 1.3.0.0 du moteur, Amazon Neptune prend en charge l'utilisation d' OpenSearch Amazon Service Serverless pour la recherche en texte intégral dans les requêtes Gremlin et SPARQL.

Rubriques

- À l'aide d'un AWS CloudFormation modèle pour démarrer Neptune-to-OpenSearch la réplication
- Activation de la recherche en texte intégral dans les bases de données Neptune existantes
- Mise à jour de l'interrogateur de flux
- Désactivation et réactivation du processus d'interrogateur de flux

À l'aide d'un AWS CloudFormation modèle pour démarrer Neptune-to-OpenSearch la réplication

Lancez un CloudFormation pile spécifique à votre région

Chacun des CloudFormation modèles ci-dessous crée une instance d'application streams-consumer dans une région spécifique AWS . Pour lancer la pile correspondante à l'aide de la CloudFormation console, cliquez sur l'un des boutons Launch Stack du tableau suivant, en fonction de la AWS région que vous souhaitez utiliser.

Région	Afficher	Afficher dans Designer	Lancer
USA Est (Virginie du Nord)	Afficher	Afficher dans Designer	Launch Stack
USA Est (Ohio)	Afficher	Afficher dans Designer	Launch Stack
USA Ouest (Californie du Nord)	Afficher	Afficher dans Designer	Launch Stack

Région	Afficher	Afficher dans Designer	Lancer
USA Ouest (Oregon)	Afficher	Afficher dans Designer	Launch Stack
Canada (Centre)	Afficher	Afficher dans Designer	Launch Stack
Amérique du Sud (São Paulo)	Afficher	Afficher dans Designer	Launch Stack
Europe (Stockholm)	Afficher	Afficher dans Designer	Launch Stack
Europe (Irlande)	Afficher	Afficher dans Designer	Launch Stack
Europe (Londres)	Afficher	Afficher dans Designer	Launch Stack
Europe (Paris)	Afficher	Afficher dans Designer	Launch Stack
Europe (Espagne)	Afficher	Afficher dans Designer	Launch Stack
Europe (Francfort)	Afficher	Afficher dans Designer	Launch Stack
Middle East (Bahrain)	Afficher	Afficher dans Designer	Launch Stack
Moyen-Orient (EAU)	Afficher	Afficher dans Designer	Launch Stack
Israël (Tel Aviv)	Afficher	Afficher dans Designer	Launch Stack

Région	Afficher	Afficher dans Designer	Lancer
Afrique (Le Cap)	Afficher	Afficher dans Designer	
Asie-Pacifique (Hong Kong)	Afficher	Afficher dans Designer	
Asie-Pacifique (Tokyo)	Afficher	Afficher dans Designer	
Asie-Pacifique (Séoul)	Afficher	Afficher dans Designer	
Asie-Pacifique (Singapour)	Afficher	Afficher dans Designer	
Asie-Pacifique (Jakarta)	Afficher	Afficher dans Designer	
Asie-Pacifique (Mumbai)	Afficher	Afficher dans Designer	
Chine (Pékin)	Afficher	Afficher dans Designer	
Chine (Ningxia)	Afficher	Afficher dans Designer	
AWS GovCloud (US-West)	Afficher	Afficher dans Designer	
AWS GovCloud (US-East)	Afficher	Afficher dans Designer	

Sur la page Créer une pile choisissez Suivant.

Ajouter des détails sur la nouvelle OpenSearch pile que vous créez

Sur la page Spécifier les détails de la pile, vous pouvez utiliser de nombreux paramètres pour contrôler la configuration de la recherche en texte intégral :

Nom de la pile : nom de la nouvelle CloudFormation pile que vous créez. Vous pouvez généralement utiliser la valeur par défaut, `NeptuneStreamPoller`.

Sous Paramètres, indiquez ce qui suit :

Configuration réseau pour le VPC sur lequel Streams Consumer est exécuté

- **`VPC`** : indiquez le nom du VPC où la fonction Lambda d'interrogation sera exécutée.
- **`List of Subnet IDs`** : sous-réseaux avec lesquels une interface réseau est établie. Ajoutez des sous-réseaux correspondant à votre cluster Neptune.
- **`List of Security Group Ids`** : fournissez les ID des groupes de sécurité qui accordent un accès entrant en écriture à votre cluster de bases de données Neptune source.
- **`List of Route Table Ids`** : nécessaire pour créer un point de terminaison Amazon DynamoDB dans votre VPC Neptune, si vous n'en avez pas déjà un. Vous devez fournir une liste séparée par des virgules des ID de table de routage associés aux sous-réseaux.
- **`Require to create Dynamo DB VPC Endpoint`** : valeur booléenne qui correspond par défaut à `true`. Vous n'avez besoin de la remplacer par `false` que si vous avez déjà créé un point de terminaison DynamoDB dans votre VPC.
- **`Require to create Monitoring VPC Endpoint`** : valeur booléenne qui correspond par défaut à `true`. Vous n'avez besoin de la modifier en `false` que si vous avez déjà créé un point de terminaison de surveillance dans votre VPC.

Interrogateur de flux

- **`Application Name`** : vous pouvez généralement conserver la valeur par défaut (`NeptuneStream`). Si vous utilisez un nom différent, il doit être unique.
- **`Memory size for Lambda Poller`** : utilisé pour définir la taille de mémoire disponible pour la fonction d'interrogateur Lambda. La valeur par défaut est 2 048 mégaoctets.
- **`Lambda Runtime`** : langage utilisé dans la fonction Lambda qui extrait les éléments du flux Neptune. Vous pouvez définir ce paramètre sur `python3.9` ou sur `java8`.

- **`S3 Bucket having Lambda code artifacts`** : laissez cette case vide, sauf si vous utilisez une fonction d'interrogation Lambda personnalisée qui se charge à partir d'un compartiment S3 différent.
- **`S3 Key corresponding to Lambda Code artifacts`** : laissez ce champ vide, sauf si vous utilisez une fonction d'interrogation Lambda personnalisée.
- **`StartingCheckpoint`** : point de contrôle de départ de l'interrogateur de flux. La valeur par défaut est `0:0`, ce qui signifie que l'interrogateur de flux doit commencer au début du flux Neptune.
- **`StreamPollerInitialState`** : état initial de l'interrogateur. La valeur par défaut est `ENABLED`, ce qui signifie que la réplication du flux débutera dès que la création complète de la pile sera terminée.
- **`Logging level for Lambda`** : en général, conservez la valeur par défaut, `INFO`.
- **`Managed Policies for Lambda Execution`** : en général, laissez ce champ vide sauf si vous utilisez une fonction d'interrogation Lambda personnalisée.
- **`Stream Records Handler`** : en général, laissez ce champ vide, sauf si vous utilisez un gestionnaire personnalisé pour les enregistrements dans les flux Neptune.
- **`Maximum records Fetched from Stream`** : vous pouvez utiliser ce paramètre pour ajuster les performances. La valeur par défaut (`100`) est un bon endroit pour commencer. Le maximum autorisé est de 10 000. Plus le nombre est élevé, moins les appels réseau sont nécessaires pour lire les enregistrements du flux, mais plus la mémoire est nécessaire pour traiter les enregistrements.
- **`Max wait time between two Polls (in Seconds)`** : détermine la fréquence à laquelle l'interrogateur Lambda est appelé pour interroger les flux Neptune. Définissez cette valeur sur 0 pour l'interrogation continue. La valeur maximale est de 3 600 secondes (1 heure). La valeur par défaut (60 secondes) est un bon point de départ, en fonction de la vitesse à laquelle vos données de graphe changent.
- **`Maximum Continuous polling period (in Seconds)`** : permet de définir un délai d'expiration pour la fonction d'interrogation Lambda. Il doit être compris entre 5 secondes et 900 secondes. La valeur par défaut (600 secondes) est un bon point de départ.
- **`Step Function Fallback Period`**— Le nombre d'unités de période de transition entre fonction et période de repli devant attendre le sondeur, après quoi la fonction d'étape est appelée via Amazon CloudWatch Events pour se remettre d'une défaillance. La valeur par défaut (5 minutes) est un bon point de départ.

- **`Step Function Fallback Period Unit`** : unités de temps utilisées pour mesurer l'élément `Step Function Fallback Period` précédent (minutes, heures, jours). La valeur par défaut (minutes) est généralement suffisante.
- **`Enable/disable creation of Elastic Search Mappings by Lambda`**— Signale la gestion enable/disable automatique des mappages Elastic Search par le stream poller. Les mappages Elastic Search sont essentiels à la conversion entre les types de données Neptune et Elastic Search. Cela ne doit être désactivé que par les utilisateurs avancés qui souhaitent contrôler directement et manuellement tous les mappages de leur index Elastic Search. La valeur par défaut est `true`.
- **`Data replication scope`**— Détermine s'il faut répliquer à la fois les nœuds et les arêtes, ou uniquement les nœuds vers OpenSearch (cela s'applique uniquement aux données du moteur Gremlin). La valeur par défaut (All) est généralement un bon endroit pour commencer.
- **`Ignore OpenSearch missing document error`**— Indicateur permettant de déterminer si une erreur liée à un document manquant OpenSearch peut être ignorée. Les erreurs de document manquant se produisent rarement, mais nécessitent une intervention manuelle si elles ne sont pas ignorées. : la valeur par défaut (`True`) est généralement un bon point de départ.
- **`Enable Non-String Indexing`** : indicateur permettant d'activer ou de désactiver l'indexation des champs sans contenu de chaîne. Si cet indicateur est défini sur`true`, les champs autres que des chaînes sont indexés OpenSearch, ou si `false` seuls les champs de chaîne sont indexés. La valeur par défaut est `true`.
- **`Properties to exclude from being inserted into OpenSearch`**— Liste séparée par des virgules de clés de propriété ou de prédicat à exclure de l'indexation. OpenSearch Si cette valeur de paramètre CFN est laissée vide, toutes les clés de propriété sont indexées.
- **`Datatypes to exclude from being inserted into OpenSearch`**— Liste séparée par des virgules de types de données de propriétés ou de prédicats à exclure de l'indexation. OpenSearch Si cette valeur de paramètre CFN est laissée vide, toutes les valeurs de propriété qui peuvent être converties en toute sécurité en OpenSearch types de données sont indexées.

Flux Neptune

- **`Endpoint of source Neptune Stream`** : (obligatoire) prend deux formes :
  - **`https://`*`your DB cluster`*`:`*`port`*`/propertygraph/stream`** (ou son alias, `https://`*`your DB cluster`*`:`*`port`*`/pg/stream`).
  - `https://`*`your DB cluster`*`:`*`port`*`/sparql/stream`
- **`Neptune Query Engine`** : choisissez Gremlin ou SPARQL.

- **`Is IAM Auth Enabled?`** : si votre cluster de bases de données Neptune utilise l'authentification IAM, définissez ce paramètre sur `true`.
- **`Neptune Cluster Resource Id`** : si votre cluster de bases de données Neptune utilise l'authentification IAM, définissez ce paramètre sur l'ID de ressource du cluster. L'ID de ressource n'est pas le même que l'ID de cluster. Elle prend plutôt la forme : `cluster-` suivi de 28 caractères alphanumériques. Vous pouvez le trouver sous Détails du cluster dans la console Neptune.

OpenSearch Cluster cible

- **`Endpoint for OpenSearch service`**— (Obligatoire) Indiquez le point de terminaison du OpenSearch service dans votre VPC.
- **`Number of Shards for OpenSearch Index`** : la valeur par défaut (5) est généralement un bon point de départ.
- **`Number of Replicas for OpenSearch Index`** : la valeur par défaut (1) est généralement un bon point de départ.
- **`Geo Location Fields for Mapping`** : si vous utilisez des champs de géolocalisation, répertoriez les clés de propriété ici.

alerte

- **`Require to create Cloud watch Alarm`**— Réglez ce `true` paramètre sur si vous souhaitez créer une CloudWatch alarme pour la nouvelle pile.
- **`SNS Topic ARN for Cloudwatch Alarm Notifications`**— L'ARN de la rubrique SNS où les notifications CloudWatch d'alarme doivent être envoyées (uniquement nécessaire si les alarmes sont activées).
- **`Email for Alarm Notifications`** : adresse e-mail à laquelle les notifications d'alarme doivent être envoyées (uniquement si les alarmes sont activées).

Pour la destination de la notification d'alarme, vous pouvez ajouter la rubrique SNS uniquement, l'adresse e-mail uniquement ou les deux.

Exécutez le CloudFormation Modèle

Vous pouvez maintenant terminer le processus de provisionnement d'une instance d'application grand public de flux Neptune comme suit :

1. Dans CloudFormation, sur la page Spécifier les détails de la pile, choisissez Next.
2. Dans la page Options, choisissez Suivant.
3. Sur la page Vérification, cochez la première case pour accepter que CloudFormation créera des ressources IAM. Cochez la deuxième case pour confirmer `CAPABILITY_AUTO_EXPAND` comme nouvelle pile.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` accepte explicitement que les macros soient étendues lors de la création de la pile, sans révision préalable. Les utilisateurs créent souvent un jeu de modifications à partir d'un modèle traité, de sorte que les modifications apportées par les macros puissent être révisées avant la création de la pile. Pour plus d'informations, consultez le fonctionnement de l' CloudFormation CreateStackAPI dans la référence des AWS CloudFormation API.

   Ensuite, choisissez Créer.

Activation de la recherche en texte intégral dans les bases de données Neptune existantes

Il s'agit des approches établies pour permettre la recherche en texte intégral dans les bases de données Amazon Neptune existantes. Selon que vous pouvez suspendre vos charges de travail d'écriture ou non, les étapes peuvent varier légèrement. Ce guide décrit les étapes recommandées pour les deux scénarios : quand vous pouvez suspendre les écritures et quand vous ne le pouvez pas. Il couvre l'activation des flux Neptune, la création d'un clone de base de données, la synchronisation des données avec un OpenSearch domaine et la configuration de mises à jour continues. Les conseils s'appuient sur des AWS services et des outils open source pour rationaliser le processus et minimiser les temps d'arrêt.

Si vous pouvez suspendre vos charges de travail d'écriture

La méthode la plus adaptée pour activer la recherche en texte intégral dans une base de données Neptune existante est généralement la suivante, à condition que vous puissiez suspendre les charges de travail d'écriture. Elle nécessite de créer un clone, d'activer les flux à l'aide d'un paramètre de cluster et de redémarrer toutes les instances. La création d'un clone étant une opération relativement rapide, le temps d'arrêt requis est limité.

Voici les étapes requises :

1. Arrêtez toutes les charges de travail d'écriture sur la base de données.
2. Activez les flux sur la base de données (voir Activation de Neptune Streams).
3. Créez un clone de la base de données (voir Clonage de base de données dans Neptune).
4. Relancez les charges de travail d'écriture.
5. Utilisez l'outil export-neptune-to-elasticsearch sur github pour effectuer une synchronisation unique entre la base de données clonée et le domaine. OpenSearch
6. Utilisez le modèle CloudFormation de votre région pour démarrer la synchronisation à partir de la base de données d'origine avec une mise à jour continue (aucune modification de configuration n'est nécessaire dans le modèle).
7. Supprimez la base de données clonée et la CloudFormation pile créée pour l'`export-neptune-to-elasticsearch`outil.

> **Note**
>
> Si vous utilisez export-neptune-to-elasticsearch avec Opensearch Serverless, la pile `CloudformationLambdaExecutionRole` créée par la pile export-neptune-to-elasticsearch doit être ajoutée à la politique d'accès aux données d'Opensearch pour que les données soient correctement répliquées.

Si vous ne pouvez pas suspendre vos charges de travail d'écriture

Si vous ne pouvez pas vous permettre de suspendre les charges de travail d'écriture sur la base de données, voici une approche qui nécessite encore moins de temps d'arrêt que celle recommandée ci-dessus, mais elle doit être appliquée avec prudence :

1. Activez les flux sur la base de données (voir Activation de Neptune Streams).
2. Créez un clone de la base de données (voir Clonage de base de données dans Neptune).
3. Obtenez les derniers `eventID` des flux de la base de données clonée en exécutant une commande de ce type sur le point de terminaison de l'API Streams (voir Appel de l'API REST Neptune Streams pour plus d'informations) :

```
curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
iteratorType=LATEST"
```

Notez les valeurs des champs `commitNum` et `opNum` dans l'objet `lastEventId` de la réponse.

4. Utilisez l'outil export-neptune-to-elasticsearch sur github pour effectuer une synchronisation unique entre la base de données clonée et le domaine. OpenSearch
5. Utilisez le modèle CloudFormation de votre région pour démarrer la synchronisation à partir de la base de données d'origine avec une mise à jour continue.

   Apportez la modification suivante lors de la création de la pile : sur la page des détails de la pile, dans la section Paramètres, définissez la valeur du `StartingCheckpoint` champ sur *`commitNum`* : *`opnum`* en utilisant les `opNum` valeurs `commitNum` et que vous avez enregistrées ci-dessus.
6. Supprimez la base de données clonée et la CloudFormation pile créée pour l'`export-neptune-to-elasticsearch`outil.

Mise à jour de l'interrogateur de flux

Les informations suivantes décrivent les étapes nécessaires pour mettre à jour le Stream Poller avec les derniers artefacts Lambda à l'aide de AWS la console de gestion.

Pour mettre à jour l'interrogateur de flux avec les derniers artefacts Lambda

Pour mettre à jour l'interrogateur de flux avec les derniers artefacts Lambda, procédez comme suit :

1. Dans le AWS Management Console, naviguez jusqu'à la CloudFormation pile parent principale CloudFormation et sélectionnez-la.
2. Sélectionnez l'option Mettre à jour correspondant à la pile.
3. Sélectionnez Remplacer le modèle actuel.
4. Pour la source du modèle, choisissez URL Amazon S3 et indiquez l'URL S3 suivante :

   ```
   https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
   neptune_to_elastic_search.json
   ```
5. Sélectionnez Next sans modifier aucun CloudFormation paramètre.
6. Choisissez Mettre à jour la pile.

La pile met à jour les artefacts Lambda avec les artefacts les plus récents.

Extension de l'interrogateur de flux pour prendre en charge les champs personnalisés

L'interrogateur de flux actuel peut facilement être étendu pour écrire du code personnalisé permettant de gérer des champs personnalisés, comme expliqué en détail dans ce billet de blog : Capture graph changes using Neptune Streams.

> **Note**
>
> Lorsque vous ajoutez un champ personnalisé OpenSearch, assurez-vous d'ajouter le nouveau champ en tant qu'objet interne d'un prédicat (voirModèle de données de recherche en texte intégral Neptune).

Désactivation et réactivation du processus d'interrogateur de flux

> **Warning**
>
> Soyez prudent lorsque vous désactivez le processus d'interrogateur de flux. Une perte de données peut se produire si le processus est suspendu au-delà de la période d'expiration des flux. La fenêtre par défaut est de 7 jours, mais à partir de la version 1.2.0.0 du moteur, vous pouvez définir une période d'expiration des flux personnalisée pouvant aller jusqu'à 90 jours.

Désactivation (interruption) du processus d'interrogateur de flux

1. Connectez-vous à la EventBridge console Amazon AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/events/.
2. Dans le volet de navigation, sélectionnez Règles.
3. Sélectionnez la règle dont le nom contient le nom que vous avez fourni comme nom d'application dans le CloudFormation modèle que vous avez utilisé pour configurer le stream poller.
4. Choisissez Désactiver.
5. Ouvrez la console Step Functions à l'adresse https://console.aws.amazon.com/states/.
6. Sélectionnez la fonction d'étape en cours d'exécution qui correspond au processus d'interrogateur de flux. Encore une fois, le nom de cette fonction d'étape contient le nom que

vous avez fourni comme nom d'application dans le CloudFormation modèle que vous avez utilisé pour configurer le stream poller. Vous pouvez effectuer un filtrage par statut d'exécution des fonctions afin de ne voir que les fonctions en cours d'exécution.

7. Choisissez Arrêter.

Re-enabling le processus Stream Poller

1. Connectez-vous à la EventBridge console Amazon AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/events/.
2. Dans le volet de navigation, sélectionnez Règles.
3. Sélectionnez la règle dont le nom contient le nom que vous avez fourni comme nom d'application dans le CloudFormation modèle que vous avez utilisé pour configurer le stream poller.
4. Sélectionnez Activer. La règle d'événement basée sur l'intervalle planifié spécifié déclenche désormais une nouvelle exécution de la fonction d'étape.

Réplication vers le mode OpenSearch sans serveur

À partir de la version 1.3.0.0 du moteur, Amazon Neptune prend en charge l'utilisation d' OpenSearch Amazon Service Serverless pour la recherche en texte intégral dans les requêtes Gremlin et SPARQL. L'utilisation de OpenSearch Serverless nécessite que vous activiez l'authentification IAM sur votre cluster de base de données Neptune. Les clusters de base de données Neptune dont l'authentification IAM est désactivée ne sont pas pris en charge avec Serverless. OpenSearch

Modifications requises pour la fonction AWS Lambda du poller

Si vous effectuez une réplication vers OpenSearch Serverless, ajoutez le rôle d'exécution du sondeur de flux Lambda à la politique d'accès aux données pour la collection Serverless. OpenSearch L'ARN du rôle d'exécution du sondeur de flux Lambda a le format suivant :

```
arn:aws:iam::(account ID):role/stack-name-NeptuneOSReplication-
NeptuneStreamPollerExecu-(uuid)
```

Modifications requises pour l'utilitaire d'importation en bloc

Si vous utilisez export-neptune-to-elasticsearchpour synchroniser des données existantes avec OpenSearch Serverless, ajoutez la politique `LambdaExecutionRole` d'accès aux données de la

CloudFormation pile pour la collection OpenSearch Serverless. L'ARN du `LambdaExecutionRole` a le format suivant :

```
arn:aws:iam::012345678901:role/stack-name-LambdaExecutionRole-(id)
```

Pour plus d'informations, consultez la section Contrôle d'accès aux données pour Amazon OpenSearch Serverless.

Modifications nécessaires au rôle IAM utilisé pour interroger Neptune

L'entité IAM (utilisateur ou rôle) utilisée pour se connecter à la base de données Neptune doit disposer d'autorisations à la fois pour Neptune et pour la collection Serverless. OpenSearch Cela signifie que votre utilisateur ou votre rôle doit disposer d'une politique OpenSearch sans serveur comme celle-ci ci-jointe :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowOpenSearchServerlessAccess",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "aoss:APIAccessAll",
      "Resource": "arn:aws:aoss:us-east-1:111122223333:collection/collection-id"
    }
  ]
}
```

Pour plus d'informations, consultez Création de déclarations de politique IAM personnalisées pour accéder aux données dans Amazon Neptune.

Interrogation depuis un OpenSearch cluster sur lequel le contrôle d'accès détaillé (FGAC) est activé

Si vous avez activé le contrôle d'accès détaillé sur votre OpenSearch cluster, vous devez également activer l'authentification IAM dans votre base de données Neptune.

L'entité IAM (utilisateur ou rôle) utilisée pour se connecter à la base de données Neptune doit disposer d'autorisations à la fois pour Neptune et pour le cluster. OpenSearch Cela signifie que votre utilisateur ou votre rôle doit être associé à une politique de OpenSearch service comme celle-ci :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFullTextSearchAccess",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "es:*",
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/domain-name/*"
    }
  ]
}
```

Pour plus d'informations, consultez Création de déclarations de politique IAM personnalisées pour accéder aux données dans Amazon Neptune.

Utilisation de la syntaxe de requête Apache Lucene dans les requêtes de recherche en texte intégral Neptune

OpenSearch prend en charge l'utilisation de la syntaxe Apache Lucene pour les requêtes query_string. Elle est particulièrement utile pour transmettre plusieurs filtres dans une même requête.

Neptune utilise une structure imbriquée pour stocker les propriétés d'un OpenSearch document (voir). Modèle de données de recherche en texte intégral Neptune Lorsque vous utilisez la syntaxe Lucene, vous devez utiliser les chemins complets vers les propriétés de ce modèle imbriqué.

Voici un exemple Gremlin :

```
g.withSideEffect("Neptune#fts.endpoint", "es_endpoint")
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V()
 .has("*", "Neptune#fts predicates.name.value:\"Jane Austin\" AND entity_type:Book")
```

Voici un exemple SPARQL :

```
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
  neptune-fts:config neptune-fts:endpoint 'http://localhost:9200 (http://
localhost:9200/)' .
  neptune-fts:config neptune-fts:queryType 'query_string' .
  neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*name.value:Ronak AND
 predicates.\\*foaf\\*surname.value:Sh*" .
  neptune-fts:config neptune-fts:field '*' .
  neptune-fts:config neptune-fts:return ?res .
}
```

Modèle de données Neptune pour les données OpenSearch

Amazon Neptune utilise une structure de document JSON unifiée pour stocker les données SPARQL et Gremlin dans Service. OpenSearch Chaque document OpenSearch correspond à une entité et stocke toutes les informations pertinentes pour cette entité. Pour Gremlin, les sommets et les arêtes sont considérés comme des entités. Les OpenSearch documents correspondants contiennent donc des informations sur les sommets, les étiquettes et les propriétés. Pour SPARQL, les sujets peuvent être considérés comme des entités, de sorte que les OpenSearch documents correspondants contiennent des informations sur toutes les paires prédicat-objet dans un seul document.

> **Note**
>
> L'implémentation de Neptune-to-OpenSearch réplication ne stocke que des données sous forme de chaîne. Toutefois, vous pouvez le modifier pour stocker d'autres types de données.

La structure de document JSON unifiée ressemble à ce qui suit.

```
{
  "entity_id": "Vertex Id/Edge Id/Subject URI",
  "entity_type": [List of Labels/rdf:type object value],
  "document_type": "vertex/edge/rdf-resource"
  "predicates": {
    "Property name or predicate URI": [
      {
        "value": "Property Value or Object Value",
        "graph": "(Only for Sparql) Named Graph Quad is present"
        "language": "(Only for Sparql) rdf:langString"
      },
      {
        "value": "Property Value 2/ Object Value 2",
      }
    ]
  }
}
```

- `entity_id` : ID unique d'entité représentant le document.
  - Pour SPARQL, il s'agit de l'URI de sujet.
  - Pour Gremlin, c'est le `Vertex_ID` ou `Edge_ID`.
- `entity_type` : représente une ou plusieurs étiquettes pour un sommet ou une arête, ou zéro ou plusieurs valeurs de prédicat `rdf:type` pour un sujet.
- `document_type` : permet de spécifier si le document actuel représente un sommet, une arête ou une ressource RDF.
- `predicates` : pour Gremlin, stocke les propriétés et les valeurs d'un sommet ou d'une arête. Pour SPARQL, il stocke les paires prédicat-objet.

  Le nom de la propriété prend le formulaire `properties.name.value` dans OpenSearch. Pour l'interroger, vous devez le nommer sous cette forme.
- `value` : valeur de propriété pour Gremlin ou valeur d'objet pour SPARQL.
- `graph` : graphe nommé pour SPARQL.
- `language` : balise de langage pour un littéral `rdf:langString` dans SPARQL.

Exemple de document SPARQL OpenSearch

Données

```
@prefix dt:   <http://example.org/datatype#> .
@prefix ex:   <http://example.org/> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

ex:simone   rdf:type     ex:Person                    ex:g1
ex:michael  rdf:type     ex:Person                    ex:g1
ex:simone   ex:likes     "spaghetti"                  ex:g1

ex:simone   ex:knows     ex:michael                   ex:g2   # Not stored in ES
ex:simone   ex:likes     "spaghetti"                  ex:g2
ex:simone   ex:status    "La vita è un sogno"@it      ex:g2

ex:simone   ex:age       "40"^^xsd:int                DG      # Not stored in ES
ex:simone   ex:dummy     "testData"^^dt:newDataType   DG      # Not stored in ES
ex:simone   ex:hates     _:bnode                              # Not stored in ES
_:bnode     ex:means     "coding"                     DG      # Not stored in ES
```

Documents

```
{
  "entity_id": "http://example.org/simone",
  "entity_type": ["http://example.org/Person"],
  "document_type": "rdf-resource"
  "predicates": {
    "http://example.org/likes": [
      {
        "value": "spaghetti",
        "graph": "http://example.org/g1"
      },
      {
        "value": "spaghetti",
        "graph": "http://example.org/g2"
      }
    ]
    "http://example.org/status": [
      {
        "value": "La vita è un sogno",
        "language": "it"       // Only present for rdf:langString
```

```
      }
    ]
  }
}
```

```
{
  "entity_id" : "http://example.org/michael",
  "entity_type" : ["http://example.org/Person"],
  "document_type": "rdf-resource"
}
```

Exemple de document Gremlin OpenSearch

Données

```
# Vertex 1
simone   label    Person       <== Label
simone   likes    "spaghetti"  <== Property
simone   likes    "rice"       <== Property
simone   age      40           <== Property

# Vertex 2
michael  label    Person       <== Label

# Edge 1
simone  knows     michael      <== Edge
e1      updated  "2019-07-03"  <== Edge Property
e1      through  "company"     <== Edge Property
e1      since     10          <== Edge Property
```

Documents

```
{
  "entity_id": "simone",
  "entity_type": ["Person"],
  "document_type": "vertex",
  "predicates": {
    "likes": [
      {
        "value": "spaghetti"
      },
      {
```

```
        "value": "rice"
      }
    ]
  }
}
```

```
{
  "entity_id" : "michael",
  "entity_type" : ["Person"],
  "document_type": "vertex"
}
```

```
{
  "entity_id": "e1",
  "entity_type": ["knows"],
  "document_type": "edge"
  "predicates": {
    "through": [
      {
        "value": "company"
      }
    ]
  }
}
```

Paramètres de recherche en texte intégral Neptune

Amazon Neptune utilise les paramètres suivants pour spécifier des OpenSearch requêtes en texte intégral dans Gremlin et SPARQL :

- **`queryType`**— (Obligatoire) Type de OpenSearch requête. (Pour une liste des types de requêtes, consultez la OpenSearch documentation). Neptune prend en charge les types de OpenSearch requêtes suivants :
  - simple_query_string : renvoie les documents en fonction d'une chaîne de requête fournie, en utilisant un analyseur avec une syntaxe Lucene limitée, mais tolérante aux pannes. Il s'agit du type de requête par défaut.

    Cette requête utilise une syntaxe simple pour analyser et diviser la chaîne de requête fournie en termes basés sur des opérateurs spéciaux. La requête analyse ensuite chaque terme indépendamment avant de renvoyer les documents correspondants.

Bien que sa syntaxe soit plus limitée que la requête `query_string`, la requête `simple_query_string` ne renvoie pas d'erreurs pour la syntaxe non valide. Au lieu de cela, il ignore toutes les parties non valides de la chaîne de requête.

- match : la requête `match` est la requête standard permettant d'effectuer une recherche en texte intégral, y compris les options de correspondance partielle.
- prefix : renvoie les documents contenant un préfixe spécifique dans un champ fourni.
- fuzzy : renvoie les documents contenant des termes similaires au terme de recherche, tels que mesurés par une distance de modification de Levenshtein.

  Une distance d'édition est le nombre de modifications d'un caractère nécessaires pour transformer un terme en un autre. Ces changements peuvent inclure :

  - Modification d'un caractère (car changé en bar).
  - Suppression d'un caractère (marre changé en mare).
  - Insertion d'un caractère (malade changé en maladie).
  - Transposition de deux caractères adjacents (chien changé en chine).

  Pour trouver des termes similaires, la requête approximative crée un ensemble de toutes les variations et extensions possibles du terme recherché à l'intérieur d'une distance d'édition spécifiée, puis renvoie des correspondances exactes pour chacune de ces variantes.
- termf : renvoie les documents qui contiennent une correspondance exacte avec un terme indiqué dans l'un des champs spécifiés.

  Vous pouvez utiliser la requête `term` pour rechercher des documents en fonction d'une valeur précise telle qu'un prix, un ID de produit ou un nom d'utilisateur.

  > ⚠ Warning
  >
  > Évitez d'utiliser la requête de terme pour les champs de texte. Par défaut, OpenSearch modifie les valeurs des champs de texte dans le cadre de son analyse, ce qui peut compliquer la recherche de correspondances exactes entre les valeurs des champs de texte.
  > Pour rechercher des valeurs de champ de texte, utilisez plutôt la requête de correspondance.
- query_string : renvoie les documents en fonction d'une chaîne de requête fournie, en utilisant un analyseur avec une syntaxe stricte (syntaxe Lucene).

Cette requête utilise une syntaxe pour analyser et diviser la chaîne de requête fournie en fonction des opérateurs, tels que AND ou NOT. La requête analyse ensuite chaque texte fractionné indépendamment avant de renvoyer les documents correspondants.

Vous pouvez utiliser la requête `query_string` pour créer une recherche complexe comprenant des caractères génériques, des recherches dans plusieurs champs, etc. Bien qu'elle soit polyvalente, la requête est stricte et renvoie une erreur si la chaîne de requête inclut une syntaxe non valide.

> **Warning**
>
> Dans la mesure où elle renvoie une erreur pour toute syntaxe non valide, nous vous déconseillons d'utiliser la requête `query_string` pour les zones de recherche. Si vous n'avez pas besoin de prendre en charge une syntaxe de requête, envisagez d'utiliser la requête `match`. Si vous avez besoin des fonctionnalités d'une syntaxe de requête, utilisez la requête `simple_query_string`, qui est moins stricte.

- **`field`**— Champ OpenSearch dans lequel effectuer la recherche. Ce paramètre ne peut être omis que si le `queryType` l'autorise (comme le font `simple_query_string` et `query_string`), auquel cas la recherche est effectuée par rapport à tous les champs. Dans Gremlin, il est implicite.

  Plusieurs champs peuvent être spécifiés si la requête l'autorise, comme le font `simple_query_string` et `query_string`.
- **`query`**— (Obligatoire) La requête à exécuter OpenSearch. Le contenu de ce champ peut varier en fonction du type de requête (queryType). Les différents types de requête acceptent des syntaxes différentes, comme le fait `Regexp` par exemple. Dans Gremlin, `query` est implicite.
- **`maxResults`** : nombre maximal de résultats à renvoyer. La valeur par défaut est le `index.max_result_window` OpenSearch paramètre, qui est lui-même défini par défaut sur 10 000. Le paramètre `maxResults` peut spécifier n'importe quel nombre inférieur à cela.

> **Important**
>
> Si vous définissez une valeur supérieure `maxResults` à cette OpenSearch `index.max_result_window` valeur et que vous essayez de récupérer plus que des `index.max_result_window` résultats, cela OpenSearch échoue avec une `Result window is too large` erreur. Cependant, Neptune gère cette situation en

douceur sans propager l'erreur. Gardez cela à l'esprit si vous essayez d'obtenir plus `index.max_result_window` résultats.

- **`minScore`** : score minimum qu'un résultat de recherche doit avoir pour être renvoyé. Consultez OpenSearchla documentation sur la pertinence pour une explication de la notation des résultats.
- **`batchSize`** : Neptune récupère toujours les données par lots (la taille par défaut du lot est de 100). Vous pouvez utiliser ce paramètre pour régler les performances. La taille du lot ne peut pas dépasser le `index.max_result_window` OpenSearch paramètre, qui est par défaut de 10 000.
- **`sortBy`**— Paramètre facultatif qui permet de trier les résultats renvoyés OpenSearch selon l'une des méthodes suivantes :
  - Champ de chaîne particulier dans le document :

    Par exemple, dans une requête SPARQL, vous pouvez spécifier :

    ```
    neptune-fts:config neptune-fts:sortBy foaf:name .
    ```

    Dans une requête Gremlin similaire, vous pouvez spécifier :

    ```
    .withSideEffect('Neptune#fts.sortBy', 'name')
    ```

  - Champ particulier autre qu'une chaîne (`long`, `double`, etc.) dans le document :

    Notez que lorsque vous effectuez un tri sur un champ autre qu'une chaîne, vous devez ajouter `.value` au nom de ce champ pour le différencier d'un champ de chaîne.

    Par exemple, dans une requête SPARQL, vous pouvez spécifier :

    ```
    neptune-fts:config neptune-fts:sortBy foaf:name.value .
    ```

    Dans une requête Gremlin similaire, vous pouvez spécifier :

    ```
    .withSideEffect('Neptune#fts.sortBy', 'name.value')
    ```

  - `score` : tri par score de correspondance (valeur par défaut).

    Si le paramètre `sortOrder` est présent mais que `sortBy` ne l'est pas, les résultats sont triés par `score` dans l'ordre spécifié par `sortOrder`.

- `id` : tri par ID, c'est-à-dire l'URI de sujet SPARQL ou l'ID de sommet ou d'arête Gremlin.

  Par exemple, dans une requête SPARQL, vous pouvez spécifier :

  ```
  neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_id' .
  ```

  Dans une requête Gremlin similaire, vous pouvez spécifier :

  ```
  .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_id')
  ```

- `label` : tri par étiquette.

  Par exemple, dans une requête SPARQL, vous pouvez spécifier :

  ```
  neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_type' .
  ```

  Dans une requête Gremlin similaire, vous pouvez spécifier :

  ```
  .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_type')
  ```

- `doc_type` : tri par type de document (SPARQL ou Gremlin).

  Par exemple, dans une requête SPARQL, vous pouvez spécifier :

  ```
  neptune-fts:config neptune-fts:sortBy 'Neptune#fts.document_type' .
  ```

  Dans une requête Gremlin similaire, vous pouvez spécifier :

  ```
  .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.document_type')
  ```

Par défaut, les OpenSearch résultats ne sont pas triés et leur ordre n'est pas déterministe, ce qui signifie qu'une même requête peut renvoyer des éléments dans un ordre différent à chaque fois qu'elle est exécutée. Pour cette raison, si le jeu de résultats est supérieur à `max_result_window`, un sous-ensemble tout à fait différent du total des résultats peut être renvoyé chaque fois qu'une requête est exécutée. Cependant, en triant, vous pouvez rendre les résultats de différentes séries plus comparables directement.

Si aucun paramètre `sortOrder` n'accompagne `sortBy`, l'ordre décroissant (`DESC`) du plus au moins grand est utilisé.

- **`sortOrder`**— Paramètre facultatif qui vous permet de spécifier si les OpenSearch résultats sont triés du plus petit au plus grand ou du plus grand au plus petit (valeur par défaut) :

  - `ASC` : ordre croissant, du moins grand au plus grand.
  - `DESC` : ordre décroissant, du plus grand au plus petit.

    Il s'agit de la valeur par défaut, utilisée lorsque le paramètre `sortBy` est présent, alors que `sortOrder` ne l'est pas.

  Si aucun des deux `sortBy` n'`sortOrder`est présent, les OpenSearch résultats ne sont pas triés par défaut.

OpenSearch Indexation sans chaîne dans Amazon Neptune

L' OpenSearch indexation sans chaîne dans Amazon Neptune permet de répliquer des valeurs autres que des chaînes pour les OpenSearch prédicats à l'aide du stream poller. Toutes les valeurs de prédicat qui peuvent être converties en toute sécurité en un OpenSearch mappage ou un type de données correspondant sont ensuite répliquées dans. OpenSearch

Pour que l'indexation sans chaîne soit activée sur une nouvelle pile, l'`Enable Non-String Indexing`indicateur du CloudFormation modèle doit être défini sur. `true` Il s'agit du paramètre par défaut. Pour mettre à jour une pile existante afin de prendre en charge l'indexation hors chaîne, consultez Mise à jour d'une pile existante ci-dessous.

> **Note**
>
> - Il est préférable de ne pas activer l'indexation hors chaîne sur les versions du moteur antérieures à **`1.0.4.2`**.
> - OpenSearch les requêtes utilisant des expressions régulières pour les noms de champs correspondant à plusieurs champs, dont certains contiennent des valeurs de chaîne et d'autres des valeurs autres que des chaînes, échouent avec une erreur. Il en va de même si les requêtes de recherche en texte intégral dans Neptune sont de ce type.
> - Lorsque vous effectuez un tri selon un champ autre qu'une chaîne, ajoutez « .value » au nom de ce champ pour le différencier d'un champ de chaîne.

Table des matières

- Mise à jour d'une pile de recherche en texte intégral Neptune pour prendre en charge l'indexation hors chaîne
- Filtrage des champs indexés dans la recherche en texte intégral Neptune
  - Filtrage par nom de propriété ou de prédicat
  - Filtrage par type de propriété ou de valeur de prédicat
- Mappage des types de données SPARQL et Gremlin vers OpenSearch
- Validation des mappages de données
- Exemples de OpenSearch requêtes autres que des chaînes dans Neptune
  - Obtenir tous les sommets de plus de 30 ans et dont le nom commence par « Si »
  - Obtenir tous les nœuds âgés de 10 à 50 ans et dont le nom a une correspondance floue avec « Ronka »
  - Obtenir tous les nœuds dont l'horodatage remonte aux 25 derniers jours
  - Obtenir tous les nœuds dont l'horodatage correspond à une année ou à un mois donnés

Mise à jour d'une pile de recherche en texte intégral Neptune pour prendre en charge l'indexation hors chaîne

Si vous utilisez déjà la recherche en texte intégral dans Neptune, voici les étapes à suivre pour prendre en charge l'indexation hors chaînes :

1. Arrêtez la fonction Lambda d'interrogateur de flux. Cette étape garantit qu'aucune nouvelle mise à jour n'est copiée lors de l'exportation. Pour ce faire, désactivez la règle d'événement cloud qui invoque cette fonction Lambda :
   - Dans le AWS Management Console, naviguez vers CloudWatch.
   - Sélectionnez Règles.
   - Choisissez la règle correspondant au nom de l'interrogateur de flux Lambda.
   - Sélectionnez Désactiver pour désactiver temporairement cette règle.
2. Supprimez l'indice Neptune actuel dans. OpenSearch Utilisez la `curl` requête suivante pour supprimer l'`amazon_neptune`index de votre OpenSearch cluster :

```
curl -X DELETE "your OpenSearch endpoint/amazon_neptune"
```

3. Lancez une exportation unique de Neptune vers. OpenSearch Il est préférable de configurer une nouvelle OpenSearch pile à ce stade, afin que les nouveaux artefacts soient collectés pour le sondeur qui effectue l'exportation.

   Suivez les étapes répertoriées ici GitHub pour démarrer l'exportation unique de vos données Neptune vers. OpenSearch

4. Mettez à jour les artefacts Lambda pour l'interrogateur de flux existant. Une fois que l'exportation des données Neptune vers OpenSearch est terminée avec succès, procédez comme suit :

   - Dans le AWS Management Console, naviguez vers CloudFormation.
   - Choisissez la CloudFormation pile parent principale.
   - Sélectionnez l'option Mettre à jour pour cette pile.
   - Sélectionnez Remplacer le modèle actuel à partir des options.
   - Pour la source du modèle, sélectionnez URL Amazon S3.
   - Pour l'URL Amazon S3, entrez :

     ```
     https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
     neptune_to_elastic_search.json
     ```

   - Choisissez Next sans modifier aucun des CloudFormation paramètres.
   - Sélectionnez Mettre à jour la pile. CloudFormation remplacera les artefacts du code Lambda pour le stream poller par les derniers artefacts.

5. Redémarrez l'interrogateur de flux. Pour ce faire, activez la CloudWatch règle appropriée :

   - Dans le AWS Management Console, naviguez vers CloudWatch.
   - Sélectionnez Règles.
   - Choisissez la règle correspondant au nom de l'interrogateur de flux Lambda.
   - Sélectionnez activer.

Filtrage des champs indexés dans la recherche en texte intégral Neptune

Les détails du CloudFormation modèle comportent deux champs qui vous permettent de spécifier des clés de propriété ou de prédicat ou des types de données à exclure de l'indexation : OpenSearch

Filtrage par nom de propriété ou de prédicat

Vous pouvez utiliser le paramètre de CloudFormation modèle facultatif nommé `Properties to exclude from being inserted into Elastic Search Index` pour fournir une liste séparée par des virgules de clés de propriété ou de prédicat à exclure de l'indexation. OpenSearch

Par exemple, supposons que vous définissiez ce paramètre sur `bob` :

```
"Properties to exclude from being inserted into Elastic Search Index" : bob
```

Dans ce cas, l'enregistrement de flux de la requête de mise à jour Gremlin suivante serait supprimé au lieu d'être intégré dans l'index :

```
g.V("1").property("bob", "test")
```

De même, vous pouvez définir le paramètre sur `http://my/example#bob` :

```
"Properties to exclude from being inserted into Elastic Search Index" : http://my/
example#bob
```

Dans ce cas, l'enregistrement de flux de la requête de mise à jour SPARQL suivante serait supprimé au lieu d'être intégré dans l'index :

```
PREFIX ex: <http://my/example#>
INSERT DATA { ex:s1 ex:bob "test"}.
```

Si vous n'entrez rien dans ce paramètre de CloudFormation modèle, toutes les clés de propriété non exclues seront indexées.

Filtrage par type de propriété ou de valeur de prédicat

Vous pouvez utiliser le paramètre de CloudFormation modèle facultatif nommé `Datatypes to exclude from being inserted into Elastic Search Index` pour fournir une liste séparée par des virgules de types de données de propriétés ou de valeurs de prédicat à exclure de l'indexation. OpenSearch

Pour SPARQL, il n'est pas nécessaire de répertorier l'URI complet du type XSD. Vous pouvez simplement répertorier le jeton du type de données. Voici les jetons de type de données valides que vous pouvez répertorier :

- string
- boolean
- float
- double
- dateTime
- date
- time
- byte
- short
- int
- long
- decimal
- integer
- nonNegativeInteger
- nonPositiveInteger
- negativeInteger
- unsignedByte
- unsignedShort
- unsignedInt
- unsignedLong

Pour Gremlin, voici les types de données valides que vous pouvez répertorier :

- string
- date
- bool
- byte
- short
- int
- long
- float

- `double`

Par exemple, supposons que vous définissiez ce paramètre sur `string` :

```
"Datatypes to exclude from being inserted into Elastic Search Index" : string
```

Dans ce cas, l'enregistrement de flux de la requête de mise à jour Gremlin suivante serait supprimé au lieu d'être intégré dans l'index :

```
g.V("1").property("myStringval", "testvalue")
```

De même, vous pouvez définir le paramètre sur `int` :

```
"Datatypes to exclude from being inserted into Elastic Search Index" : int
```

Dans ce cas, l'enregistrement de flux de la requête de mise à jour SPARQL suivante serait supprimé au lieu d'être intégré dans l'index :

```
PREFIX ex: <http://my/example#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
INSERT DATA { ex:s1 ex:bob "11"^^xsd:int }.
```

Si vous n'entrez rien dans ce paramètre de CloudFormation modèle, toutes les propriétés dont les valeurs peuvent être converties en toute sécurité en OpenSearch équivalents seront indexées. Les types répertoriés qui ne sont pas pris en charge par le langage de requête seront ignorés.

Mappage des types de données SPARQL et Gremlin vers OpenSearch

Les nouveaux mappages de types de données OpenSearch sont créés en fonction du type de données utilisé dans la propriété ou l'objet. Commen certains champs comprennent des valeurs de différents types, le mappage initial peut exclure certaines valeurs du champ.

Les types de données Neptune correspondent aux OpenSearch types de données comme suit :

Types SPARQL	Types Gremlin	OpenSearch types
`XSD:int`	`byte`	`long`
`XSD:unsignedInt`	`short`	

Types SPARQL	Types Gremlin	OpenSearch types
XSD:integer XSD:byte XSD:unsignedByte XSD:short XSD:unsignedShort XSD:long XSD:unsignedLong	int long	
XSD:float XSD:double XSD:decimal	float double	double
XSD:boolean	bool	boolean
XSD:datetime XSD:date	date	date
XSD:string XSD:time	string	text
Type de données personnalisé	S/O	text
Tout autre type de données	S/O	text

Par exemple, la requête de mise à jour Gremlin suivante entraîne l'ajout d'un nouveau mappage pour « NewField » OpenSearch, à savoir : `{ "type" : "double" }`

```
g.V("1").property("newField" 10.5)
```

De même, la requête de mise à jour SPARQL suivante entraîne l'ajout d'un nouveau mappage pour « ex:byte », à OpenSearch savoir : `{ "type" : "long" }`

```
PREFIX ex: <http://my/example#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>

INSERT DATA { ex:test ex:byte "123"^^xsd:byte }.
```

> **Note**
>
> Comme vous pouvez le constater, un élément mappé depuis Neptune OpenSearch vers peut se retrouver avec un type de données différent de celui qu'il possède OpenSearch dans Neptune. Cependant, il existe un champ de texte explicite dans OpenSearch « datatype » qui enregistre le type de données que possède l'élément dans Neptune.

Validation des mappages de données

Les données sont répliquées OpenSearch depuis Neptune selon le processus suivant :

- Si un mappage pour le champ en question est déjà présent dans OpenSearch :
  - Si les données peuvent être converties en toute sécurité dans le mappage existant à l'aide des règles de validation des données, stockez le champ dans OpenSearch.
  - Si ce n'est pas le cas, abandonnez l'enregistrement de mise à jour du flux correspondant.
- S'il n'existe aucun mappage pour le champ en question, recherchez un type de OpenSearch données correspondant au type de données du champ dans Neptune.
  - Si les données de champ peuvent être converties en toute sécurité au OpenSearch type de données à l'aide des règles de validation des données, stockez le nouveau mappage et les nouvelles données de champ dans. OpenSearch
  - Si ce n'est pas le cas, abandonnez l'enregistrement de mise à jour du flux correspondant.

Les valeurs sont validées par rapport à des OpenSearch types équivalents ou à OpenSearch des mappages existants plutôt qu'aux types Neptune. Par exemple, la validation de la valeur `"123"` dans `"123"^^xsd:int` est effectuée par rapport au type `long` plutôt qu'au type `int`.

Bien que Neptune essaie de répliquer toutes les données, dans certains cas OpenSearch, les types de données OpenSearch sont totalement différents de ceux de Neptune, et dans de tels cas, les enregistrements sont ignorés au lieu d'être indexés. OpenSearch

Par exemple, dans Neptune, une propriété peut avoir plusieurs valeurs de types différents, alors que dans OpenSearch un champ, elle doit avoir le même type sur l'ensemble de l'index.

En activant les journaux de débogage, vous pouvez voir quels enregistrements ont été supprimés lors de l'exportation de OpenSearch Neptune vers. Voici un exemple d'entrée du journal de débogage :

```
Dropping Record : Data type not a valid Gremlin type
<Record>
```

Les types de données sont validés comme suit :

- **`text`**— Toutes les valeurs de Neptune peuvent être mappées en toute sécurité au texte saisi. OpenSearch
- **`long`**— Les règles suivantes pour les types de données Neptune s'appliquent lorsque le type de OpenSearch mappage est long (dans les exemples ci-dessous, on suppose qu'il `"testLong"` possède un `long` type de mappage) :
  - `boolean` : non valide, ne peut pas être converti, et l'enregistrement de mise à jour du flux correspondant est supprimé.

    Les exemples de G705 non valides sont les suivants :

    ```
    "testLong" : true.
    "testLong" : false.
    ```

    Voici des exemples de SPARQL non valides :

    ```
    ":testLong" : "true"^^xsd:boolean
    ":testLong" : "false"^^xsd:boolean
    ```

  - `datetime` : non valide, ne peut pas être converti, et l'enregistrement de mise à jour du flux correspondant est supprimé.

    Voici un exemple de G705 non valide :

    ```
    ":testLong" :  datetime('2018-11-04T00:00:00').
    ```

Voici un exemple de SPARQL non valide :

```
":testLong" : "2016-01-01"^^xsd:date
```

- `floatdouble`, ou `decimal` — Si la valeur dans Neptune est un entier pouvant contenir 64 bits, elle est valide et stockée sous forme longue, mais si elle comporte une partie fractionnaire, est a ou an, ou est supérieure à 9 223 372 036 854 775 807 ou inférieure à -9,223 372 036 854 775 808`INF`, elle n'est pas valide et l'enregistrement de mise à jour du flux correspondant est supprimé. OpenSearch `NaN`

  Les exemples valides de G705 sont les suivants :

```
"testLong" :  145.0.
":testLong" :  123
":testLong" :  -9223372036854775807
```

  Voici des exemples de SPARQL valides :

```
":testLong" : "145.0"^^xsd:float
":testLong" :  145.0
":testLong" : "145.0"^^xsd:double
":testLong" : "145.0"^^xsd:decimal
":testLong" : "-9223372036854775807"
```

  Les exemples de G705 non valides sont les suivants :

```
"testLong" :  123.45
":testLong" :  9223372036854775900
```

  Voici des exemples de SPARQL non valides :

```
":testLong" :  123.45
":testLong" :  9223372036854775900
":testLong" : "123.45"^^xsd:float
":testLong" : "123.45"^^xsd:double
":testLong" : "123.45"^^xsd:decimal
```

- `string`— Si la valeur dans Neptune est une représentation sous forme de chaîne d'un entier pouvant être contenu dans un entier de 64 bits, elle est valide et est convertie en entrée.

`long` OpenSearch Toute autre valeur de chaîne n'est pas valide pour un `long` mappage Elasticseearch, et l'enregistrement de mise à jour du flux correspondant est supprimé.

Les exemples valides de G705 sont les suivants :

```
"testLong" :  "123".
":testLong" :  "145.0"
":testLong" :  "-9223372036854775807"
```

Voici des exemples de SPARQL valides :

```
":testLong" : "145.0"^^xsd:string
":testLong" : "-9223372036854775807"^^xsd:string
```

Les exemples de G705 non valides sont les suivants :

```
"testLong" :  "123.45"
":testLong" :  "9223372036854775900"
":testLong" :  "abc"
```

Voici des exemples de SPARQL non valides :

```
":testLong" : "123.45"^^xsd:string
":testLong" : "abc"
":testLong" : "9223372036854775900"^^xsd:string
```

- **`double`**— Si le type de OpenSearch mappage est le cas`double`, les règles suivantes s'appliquent (ici, le champ « TestDouble » est supposé contenir un `double` mappage OpenSearch) :
  - `boolean` : non valide, ne peut pas être converti, et l'enregistrement de mise à jour du flux correspondant est supprimé.

    Les exemples de G705 non valides sont les suivants :

    ```
    "testDouble" : true.
    "testDouble" : false.
    ```

    Voici des exemples de SPARQL non valides :

    ```
    ":testDouble" : "true"^^xsd:boolean
    ```

```
  ":testDouble" : "false"^^xsd:boolean
```

- `datetime` : non valide, ne peut pas être converti, et l'enregistrement de mise à jour du flux correspondant est supprimé.

  Voici un exemple de G705 non valide :

  ```
    ":testDouble" :  datetime('2018-11-04T00:00:00').
  ```

  Voici un exemple de SPARQL non valide :

  ```
    ":testDouble" : "2016-01-01"^^xsd:date
  ```

- Virgule flottante `NaN` ou `INF` — Si la valeur dans SPARQL est une valeur à virgule flottante `NaN` ou`INF`, elle n'est pas valide et l'enregistrement de mise à jour du flux correspondant est supprimé.

  Voici des exemples de SPARQL non valides :

  ```
  "  :testDouble" : "NaN"^^xsd:float
    ":testDouble" : "NaN"^^double
    ":testDouble" : "INF"^^double
    ":testDouble" : "-INF"^^double
  ```

- nombre ou chaîne numérique — Si la valeur dans Neptune est une autre représentation numérique ou sous forme de chaîne numérique d'un nombre qui peut être exprimé en toute sécurité sous la forme d'`double`un a, alors elle est valide et est convertie en entrée. `double` OpenSearch Toute autre valeur de chaîne n'est pas valide pour un OpenSearch `double` mappage et l'enregistrement de mise à jour du flux correspondant est supprimé.

  Les exemples valides de G705 sont les suivants :

  ```
    "testDouble" :  123
    ":testDouble" :  "123"
    ":testDouble" :  145.67
    ":testDouble" :  "145.67"
  ```

  Voici des exemples de SPARQL valides :

  ```
    ":testDouble" :  123.45
  ```

```
  ":testDouble" :  145.0
  ":testDouble" : "123.45"^^xsd:float
  ":testDouble" : "123.45"^^xsd:double
  ":testDouble" : "123.45"^^xsd:decimal
  ":testDouble" : "123.45"^^xsd:string
```

Voici un exemple de G705 non valide :

```
  ":testDouble" :  "abc"
```

Voici un exemple de SPARQL non valide :

```
  ":testDouble" : "abc"
```

- **`date`**— Si le type de OpenSearch mappage est`date`, Neptune `date` et sa `dateTime` valeur sont valides, de même que toute valeur de chaîne pouvant être analysée avec succès selon un format. `dateTime`

  Les exemples valides dans G705 ou SPARQL sont les suivants :

```
Date(2016-01-01)
"2016-01-01" "
2003-09-25T10:49:41"
"2003-09-25T10:49"
"2003-09-25T10"
"20030925T104941-0300"
"20030925T104941"
"2003-Sep-25" "
Sep-25-2003"
"2003.Sep.25"
"2003/09/25"
"2003 Sep 25" "
Wed, July 10, '96"
"Tuesday, April 12, 1952 AD 3:30:42pm PST"
"123"
"-123"
"0"
"-0"
"123.00"
"-123.00"
```

Les exemples suivants ne sont pas valides :

```
  123.45
  True
  "abc"
```

Exemples de OpenSearch requêtes autres que des chaînes dans Neptune

Neptune ne prend actuellement pas directement en charge les requêtes OpenSearch de plage. Cependant, vous pouvez obtenir le même effet en utilisant la syntaxe Lucene et query-type="query_string", comme vous pouvez le voir dans les exemples de requêtes suivants.

Obtenir tous les sommets de plus de 30 ans et dont le nom commence par « Si »

Dans Gremlin :

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.age.value:>30 && predicates.name.value:Si*');
```

En SPARQL :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*age.value:>30 AND
 predicates.\\*foaf\\*name.value:Si*" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Dans ce cas, "`\\*foaf\\*age` est utilisé à la place de l'URI complet par souci de concision. Cette expression régulière récupère tous les champs contenant `foaf` et `age` dans l'URI.

Obtenir tous les nœuds âgés de 10 à 50 ans et dont le nom a une correspondance floue avec « Ronka »

Dans Gremlin :

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.age.value:[10 TO 50] AND
 predicates.name.value:Ronka~');
```

En SPARQL :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*age.value:[10 TO 50] AND
 predicates.\\*foaf\\*name.value:Ronka~" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Obtenir tous les nœuds dont l'horodatage remonte aux 25 derniers jours

Dans Gremlin :

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:>now-25d');
```

En SPARQL :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
```

```
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\
\*timestamp.value:>now-25d~" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Obtenir tous les nœuds dont l'horodatage correspond à une année ou à un mois donnés

Dans Gremlin, avec des expressions mathématiques de date dans la syntaxe de Lucene, pour décembre 2020 :

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:>2020-12');
```

Alternative Gremlin :

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:[2020-12 TO 2021-01]');
```

Full-text-search exécution de requêtes dans Amazon Neptune

Dans une requête incluant full-text-search, Neptune essaie de placer les full-text-search appels en premier, avant les autres parties de la requête. Cela permet de réduire le nombre d'appels OpenSearch et, dans la plupart des cas, d'améliorer considérablement les performances. Cependant, il ne s'agit en aucun cas d'une hard-and-fast règle. Dans certaines situations, par exemple, un élément `PatternNode` ou `UnionNode` peut précéder un appel de recherche en texte intégral.

Prenons la requête Gremlin suivante vers une base de données contenant 100 000 instances de `Person` :

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .hasLabel('Person')
 .has('name', 'Neptune#fts marcello~');
```

Si cette requête était exécutée dans l'ordre dans lequel les étapes apparaissent, 100 000 solutions arriveraient OpenSearch, provoquant des centaines d' OpenSearch appels. En fait, Neptune appelle

d' OpenSearch abord, puis joint les résultats aux résultats de Neptune. Dans la plupart des cas, cela est beaucoup plus rapide que l'exécution de la requête dans l'ordre d'origine.

Vous pouvez empêcher cette réorganisation de l'exécution de l'étape de requête à l'aide de l'indicateur de requête noReordering :

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .withSideEffect('Neptune#noReordering', true)
 .hasLabel('Person')
 .has('name', 'Neptune#fts marcello~');
```

Dans ce second cas, l'étape `.hasLabel` est exécutée en premier et l'étape `.has('name', 'Neptune#fts marcello~')` en deuxième.

Pour un autre exemple, prenez une requête SPARQL par rapport au même type de données :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT ?person WHERE {
  ?person rdf:type foaf:Person .
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mike' .
    neptune-fts:config neptune-fts:return ?person .
  }
}
```

Ici encore, Neptune exécute d'abord la partie `SERVICE` de la requête, puis joint les résultats aux données `Person`. Vous pouvez supprimer ce comportement en utilisant l'indicateur de requête joinOrder :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?person WHERE {
  hint:Query hint:joinOrder "Ordered" .
  ?person rdf:type foaf:Person .
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:field foaf:name .
```

```
    neptune-fts:config neptune-fts:query 'mike' .
    neptune-fts:config neptune-fts:return ?person .
  }
}
```

De nouveau, dans la deuxième requête, les parties sont exécutées dans l'ordre où elles apparaissent dans la requête.

> **Note**
>
> Interroger un alias OpenSearch sur un index, au lieu d'interroger directement un index OpenSearch, peut produire des résultats incorrects. Vous devez interroger directement l'index opensearch et non l'alias.

Exemples de requêtes SPARQL utilisant la recherche en texte intégral dans Neptune

Voici quelques exemples de requêtes SPARQL qui utilisent la recherche en texte intégral dans Amazon Neptune.

Exemple de requête de correspondance SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'match' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'michael' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemple de requête de préfixe SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
```

```
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'prefix' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mich' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemple de requête partielle SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'fuzzy' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mikael' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemple de requête de terme SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'term' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'Dr. Kunal' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemple de requête query_string SPARQL

Cette requête spécifie plusieurs champs.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ OR rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:field foaf:surname .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemple de requête simple_query_string SPARQL

La requête suivante spécifie les champs utilisant le caractère générique (« * »).

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'simple_query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemple de requête de tri par champ de chaîne SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy foaf:name .
```

```
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemple de requête de de tri de champ hors chaîne SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name.value .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy dc:date.value .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemple de requête de tri par ID SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_id' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemple de requête de tri par étiquette SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
```

```
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_type' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemple de requête de tri par doc_type SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.document_type' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Exemple d'utilisation de la syntaxe Lucene en SPARQL

La syntaxe Lucene n'est prise en charge que pour les `query_string` requêtes dans. OpenSearch

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'predicates.\\foaf\\name.value:micheal AND
 predicates.\\foaf\\surname.value:sh' .
    neptune-fts:config neptune-fts:field ‘’ .
    neptune-fts:config neptune-fts:return ?res .
```

```
  }
}
```

Utilisation de la recherche en texte intégral Neptune dans des requêtes Gremlin

`NeptuneSearchStep` permet les requêtes de recherche en texte intégral pour la partie d'une traversée Gremlin qui n'est pas convertie en étapes Neptune. Prenons l'exemple d'une requête similaire à la suivante.

```
g.withSideEffect("Neptune#fts.endpoint", "your-es-endpoint-URL")
  .V()
      .tail(100)
      .has("name", "Neptune#fts mark*")           <== # Limit the search on name
```

Cette requête est convertie en traversée optimisée suivante dans Neptune.

```
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=4}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [NeptuneTailGlobalStep(100),
 NeptuneTinkerpopTraverserConverterStep, NeptuneSearchStep {
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=your-OpenSearch-endpoint-URL}
    }
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=your-OpenSearch-endpoint-URL}
    }
}]
```

Les exemples suivants sont des requêtes Gremlin par rapport aux données sur les routes aériennes :

Requête de base Gremlin **match** insensible à la casse

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'match')
  .V().has("city","Neptune#fts dallas")

==>v[186]
==>v[8]
```

Requête Gremlin **match**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'match')
  .V().has("city","Neptune#fts southampton")
     .local(values('code','city').fold())
     .limit(5)

==>[SOU, Southampton]
```

Requête Gremlin **fuzzy**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has("city","Neptune#fts allas~").values('city').limit(5)

==>Dallas
==>Dallas
==>Walla Walla
==>Velas
==>Altai
```

Requête approximative Gremlin **query_string**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has("city","Neptune#fts allas~").values('city').limit(5)
```

```
==>Dallas
==>Dallas
```

Requête d'expression régulière Gremlin **query_string**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has("city","Neptune#fts /[dp]allas/").values('city').limit(5)

==>Dallas
==>Dallas
```

Requête hybride Gremlin

Cette requête utilise un index interne Neptune et l' OpenSearch index de la même requête.

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has("region","GB-ENG")
      .has('city','Neptune#fts L*')
      .values('city')
      .dedup()
      .limit(10)

==>London
==>Leeds
==>Liverpool
==>Land's End
```

Exemple de recherche en texte intégral simple Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has('desc','Neptune#fts regional municipal')
      .local(values('code','desc').fold())
      .limit(100)

==>[HYA, Barnstable Municipal Boardman Polando Field]
```

```
==>[SPS, Sheppard Air Force Base-Wichita Falls Municipal Airport]
==>[ABR, Aberdeen Regional Airport]
==>[SLK, Adirondack Regional Airport]
==>[BFD, Bradford Regional Airport]
==>[EAR, Kearney Regional Airport]
==>[ROT, Rotorua Regional Airport]
==>[YHD, Dryden Regional Airport]
==>[TEX, Telluride Regional Airport]
==>[WOL, Illawarra Regional Airport]
==>[TUP, Tupelo Regional Airport]
==>[COU, Columbia Regional Airport]
==>[MHK, Manhattan Regional Airport]
==>[BJI, Bemidji Regional Airport]
==>[HAS, Hail Regional Airport]
==>[ALO, Waterloo Regional Airport]
==>[SHV, Shreveport Regional Airport]
==>[ABI, Abilene Regional Airport]
==>[GIZ, Jizan Regional Airport]
==>[USA, Concord Regional Airport]
==>[JMS, Jamestown Regional Airport]
==>[COS, City of Colorado Springs Municipal Airport]
==>[PKB, Mid Ohio Valley Regional Airport]
```

Requête Gremlin utilisant **`query_string`** avec les opérateurs '+' et '-'

Bien que le type de requête `query_string` soit beaucoup moins indulgent que le type `simple_query_string` par défaut, il permet des requêtes plus précises. La première requête ci-dessous utilise `query_string`, tandis que la seconde utilise la valeur par défaut `simple_query_string` :

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
 . V().has('desc','Neptune#fts +London -(Stansted|Gatwick)')
      .local(values('code','desc').fold())
      .limit(10)

==>[LHR, London Heathrow]
==>[YXU, London Airport]
==>[LTN, London Luton Airport]
==>[SEN, London Southend Airport]
==>[LCY, London City Airport]
```

Notez comment `simple_query_string` dans les exemples ci-dessous ignore tranquillement les opérateurs « + » et « - » :

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has('desc','Neptune#fts +London -(Stansted|Gatwick)')
      .local(values('code','desc').fold())
      .limit(10)

==>[LHR, London Heathrow]
==>[YXU, London Airport]
==>[LGW, London Gatwick]
==>[STN, London Stansted Airport]
==>[LTN, London Luton Airport]
==>[SEN, London Southend Airport]
==>[LCY, London City Airport]
==>[SKG, Thessaloniki Macedonia International Airport]
==>[ADB, Adnan Menderes International Airport]
==>[BTV, Burlington International Airport]
```

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has('desc','Neptune#fts +(regional|municipal) -(international|bradford)')
      .local(values('code','desc').fold())
      .limit(10)

==>[CZH, Corozal Municipal Airport]
==>[MMU, Morristown Municipal Airport]
==>[YBR, Brandon Municipal Airport]
==>[RDD, Redding Municipal Airport]
==>[VIS, Visalia Municipal Airport]
==>[AIA, Alliance Municipal Airport]
==>[CDR, Chadron Municipal Airport]
==>[CVN, Clovis Municipal Airport]
==>[SDY, Sidney Richland Municipal Airport]
==>[SGU, St George Municipal Airport]
```

Requête Gremlin **`query_string`** avec les opérateurs **`AND`** et **`OR`**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
```

```
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has('desc','Neptune#fts (St AND George) OR (St AND Augustin)')
      .local(values('code','desc').fold())
      .limit(10)

==>[YIF, St Augustin Airport]
==>[STG, St George Airport]
==>[SGO, St George Airport]
==>[SGU, St George Municipal Airport]
```

Requête Gremlin **`term`**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'term')
  .V().has("SKU","Neptune#fts ABC123DEF9")
      .local(values('code','city').fold())
      .limit(5)

==>[AUS, Austin]
```

Requête Gremlin **`prefix`**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'prefix')
  .V().has("icao","Neptune#fts ka")
      .local(values('code','icao','city').fold())
      .limit(5)

==>[AZO, KAZO, Kalamazoo]
==>[APN, KAPN, Alpena]
==>[ACK, KACK, Nantucket]
==>[ALO, KALO, Waterloo]
==>[ABI, KABI, Abilene]
```

Utilisation de la syntaxe Lucene dans Neptune Gremlin

Dans Neptune Gremlin, vous pouvez également écrire des requêtes très puissantes en utilisant la syntaxe de requête Lucene. Notez que la syntaxe Lucene n'est prise en charge que pour les `query_string` requêtes dans. OpenSearch

Supposons les données suivantes :

```
g.addV("person")
        .property(T.id, "p1")
        .property("name", "simone")
        .property("surname", "rondelli")

g.addV("person")
        .property(T.id, "p2")
        .property("name", "simone")
        .property("surname", "sengupta")

g.addV("developer")
        .property(T.id, "p3")
        .property("name", "simone")
        .property("surname", "rondelli")
```

En utilisant la syntaxe Lucene, qui est appelée lorsque `queryType` est `query_string`, vous pouvez rechercher ces données par nom et prénom comme suit :

```
g.withSideEffect("Neptune#fts.endpoint", "es_endpoint")
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts predicates.name.value:simone AND
 predicates.surname.value:rondelli")

==> v[p1], v[p3]
```

Notez qu'au cours de l'étape `has()` ci-dessus, le champ est remplacé par `"*"`. En fait, toute valeur placée là est remplacée par les champs auxquels vous accédez dans la requête. Vous accédez au champ de nom en utilisant `predicates.name.value,`, car le modèle de données est structuré de cette façon.

Vous pouvez effectuer une recherche par nom, prénom et étiquette, comme suit :

```
g.withSideEffect("Neptune#fts.endpoint", getEsEndpoint())
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts predicates.name.value:simone AND
 predicates.surname.value:rondelli AND entity_type:person")
```

```
==> v[p1]
```

L'étiquette est accessible en utilisant `entity_type`, encore une fois parce que le modèle de données est structuré de cette façon.

Vous pouvez également inclure des conditions d'imbrication :

```
g.withSideEffect("Neptune#fts.endpoint", getEsEndpoint())
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts (predicates.name.value:simone AND
 predicates.surname.value:rondelli AND entity_type:person) OR
 predicates.surname.value:sengupta")

==> v[p1], v[p2]
```

Insertion d'un graphe moderne TinkerPop

```
g.addV('person').property(T.id, '1').property('name', 'marko').property('age', 29)
 .addV('personr').property(T.id, '2').property('name', 'vadas').property('age', 27)
 .addV('software').property(T.id, '3').property('name', 'lop').property('lang', 'java')
 .addV('person').property(T.id, '4').property('name', 'josh').property('age', 32)
 .addV('software').property(T.id, '5').property('name', 'ripple').property('lang',
 'java')
 .addV('person').property(T.id, '6').property('name', 'peter').property('age', 35)

g.V('1').as('a').V('2').as('b').addE('knows').from('a').to('b').property('weight',
 0.5f).property(T.id, '7')
 .V('1').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.4f).property(T.id, '9')
 .V('4').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.4f).property(T.id, '11')
 .V('4').as('a').V('5').as('b').addE('created').from('a').to('b').property('weight',
 1.0f).property(T.id, '10')
 .V('6').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.2f).property(T.id, '12')
 .V('1').as('a').V('4').as('b').addE('knows').from('a').to('b').property('weight',
 1.0f).property(T.id, '8')
```

Exemple de valeur de tri par champ de chaîne

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
```

```
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .withSideEffect('Neptune#fts.sortOrder', 'asc')
  .withSideEffect('Neptune#fts.sortBy', 'name')
  .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Exemple de valeur de tri par champ hors chaîne

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .withSideEffect('Neptune#fts.sortOrder', 'asc')
  .withSideEffect('Neptune#fts.sortBy', 'age.value')
  .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Exemple de valeur de tri par champ d'ID

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
.withSideEffect('Neptune#fts.queryType', 'query_string')
.withSideEffect('Neptune#fts.sortOrder', 'asc')
.withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_id')
.V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Exemple de valeur de tri par champ d'étiquette

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .withSideEffect('Neptune#fts.sortOrder', 'asc')
  .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_type')
  .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Exemple de valeur de tri par champ **`document_type`**

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .withSideEffect('Neptune#fts.sortOrder', 'asc')
  .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.document_type')
  .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Résolution des problèmes liés à la recherche Neptune en texte intégral

> **Note**
>
> Si vous avez activé le contrôle d'accès détaillé sur votre OpenSearch cluster, vous devez également activer l'authentification IAM dans votre base de données Neptune.

Pour diagnostiquer les problèmes liés à la réplication depuis Neptune vers OpenSearch, consultez les CloudWatch journaux de la fonction Lambda de votre sondeur. Ces journaux fournissent des informations détaillées sur le nombre d'enregistrements lus dans le flux et sur le nombre d'enregistrements répliqués avec succès. OpenSearch

Vous pouvez également modifier le niveau LOGGING de la fonction Lambda en modifiant la variable d'environnement `LoggingLevel`.

> **Note**
>
> Lorsque cette option est `LoggingLevel` définie sur`DEBUG`, vous pouvez afficher des informations supplémentaires, telles que les enregistrements de flux supprimés et la raison pour laquelle chacun a été supprimé, tout en répliquant les données StreamPoller de OpenSearch Neptune vers. Cette approche peut être utile si vous constatez qu'il vous manque des enregistrements.

L'application Neptune Streams destinée aux consommateurs publie deux indicateurs CloudWatch qui peuvent également vous aider à diagnostiquer les problèmes :

- `StreamRecordsProcessed` : nombre d'enregistrements traités par l'application par unité de temps. Utile pour suivre le taux d'exécution de l'application.
- `StreamLagTime` : différence de temps, en millisecondes, entre l'heure actuelle et l'heure de validation d'un enregistrement de flux en cours de traitement. Cette métrique montre à quel point l'application destinée aux consommateurs est en retard.

En outre, toutes les métriques liées au processus de réplication sont exposées dans un tableau de bord portant le même nom que celui `ApplicationName` fourni lorsque vous avez instancié l'application à l'aide du CloudWatch modèle. CloudWatch

Vous pouvez également choisir de créer une CloudWatch alarme qui se déclenche chaque fois que le sondage échoue plus de deux fois de suite. Pour ce faire, définissez le champ `CreateCloudWatchAlarm` sur `true` lorsque vous instanciez l'application. Indiquez ensuite les adresses e-mail auxquelles vous souhaitez recevoir une notification lorsque l'alarme est déclenchée.

Dépannage d'un processus qui échoue lors de la lecture d'enregistrements à partir du flux

Si un processus échoue lors de la lecture d'enregistrements à partir du flux, assurez-vous que vous disposez des éléments suivants :

- Le flux est activé sur votre cluster.
- Le point de terminaison de flux Neptune est au format correct :
  - Pour Gremlin ou openCypher : `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/propertygraph/stream` ou son alias, `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/pg/stream`
  - Pour SPARQL : `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/sparql/stream`
- Le point de terminaison DynamoDB est configuré pour votre VPC.
- Le point de terminaison de surveillance est configuré pour vos sous-réseaux VPC.

Résolution des problèmes liés à l'échec d'un processus lors de l'écriture de données dans OpenSearch

Si un processus échoue lors de l'écriture d'enregistrements dans OpenSearch, assurez-vous que vous disposez des éléments suivants :

- Votre version d'Elasticsearch est 7.1 ou supérieure, ou Opensearch 2.3 ou version ultérieure.
- OpenSearch est accessible depuis la fonction Poller Lambda de votre VPC.
- La politique de sécurité attachée à OpenSearch autorise les HTTP/HTTPS demandes entrantes.

Résolution des out-of-sync problèmes entre Neptune et une configuration OpenSearch de réplication existante

Vous pouvez suivre les étapes ci-dessous pour synchroniser à nouveau une base de données et un OpenSearch domaine Neptune avec les données les plus récentes en cas de out-of-sync problèmes entre eux résultant d'une corruption de données `ExpiredStreamException` ou d'une corruption de données.

Notez que cette approche supprime toutes les données du OpenSearch domaine et les resynchronise à partir de l'état actuel de la base de données Neptune. Aucune donnée n'a donc besoin d'être rechargée dans la base de données Neptune.

1. Désactivez le processus de réplication comme décrit dans Désactivation (suspension) du processus d'interrogateur de flux.
2. Supprimez l'index Neptune du OpenSearch domaine à l'aide de la commande suivante :

   ```
   curl -X DELETE "(your OpenSearch endpoint)/amazon_neptune"
   ```

3. Créez un clone de la base de données (voir Clonage de base de données dans Neptune).
4. Obtenez les derniers `eventID` des flux de la base de données clonée en exécutant une commande de ce type sur le point de terminaison de l'API Streams (voir Appel de l'API REST Neptune Streams pour plus d'informations) :

   ```
   curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
   iteratorType=LATEST"
   ```

   Notez les valeurs des champs `commitNum` et `opNum` dans l'objet `lastEventId` de la réponse.
5. Utilisez l'export-neptune-to-elasticsearchoutil sur github pour effectuer une synchronisation unique entre la base de données clonée et le OpenSearch domaine.
6. Accédez à la table DynamoDB correspondant à la pile de réplication. Le nom de la table sera le nom de l'application que vous avez spécifié dans le CloudFormation modèle (le nom par défaut est`NeptuneStream`) avec un `-LeaseTable` suffixe. En d'autres termes, le nom de la table par défaut sera `NeptuneStream-LeaseTable`.

   Vous pouvez explorer les lignes de la table en les analysant, car il ne devrait y avoir qu'une seule ligne. Apportez les modifications suivantes en utilisant les valeurs `commitNum` et `opNum` que vous avez enregistrées ci-dessus :

- Remplacez la valeur du champ `checkpoint` dans la table par la valeur dont vous avez pris note pour `commitNum`.
- Remplacez la valeur du champ `checkpointSubSequenceNumber` dans la table par la valeur dont vous avez pris note pour `opNum`.

7. Réactivez le processus de réplication comme décrit dans Réactivation du processus d'interrogateur de flux.
8. Supprimez la base de données clonée et la CloudFormation pile créée pour l'`export-neptune-to-elasticsearch`outil.

Utilisation AWS Lambda des fonctions dans Amazon Neptune

AWS Lambda les fonctions ont de nombreuses utilisations dans les applications Amazon Neptune. Nous fournissons ici des conseils généraux sur l'utilisation des fonctions Lambda avec l'un des pilotes et variantes de langage Greminda les plus courants, ainsi que des exemples spécifiques de fonctions Lambda écrites en Java et Python. JavaScript

> **Note**
>
> La meilleure façon d'utiliser les fonctions Lambda avec Neptune a changé avec les récentes versions du moteur. Neptune avait l'habitude de laisser les connexions inactives ouvertes longtemps après le recyclage d'un contexte d'exécution Lambda, ce qui pouvait entraîner une fuite de ressources sur le serveur. Pour pallier ce problème, nous avions l'habitude de recommander l'ouverture et la fermeture d'une connexion à chaque invocation Lambda. À partir de la version 1.0.3.0 du moteur, le délai d'inactivité des connexions a toutefois été réduit afin que les connexions ne provoquent plus de fuite après le recyclage d'un contexte d'exécution Lambda inactif. Nous recommandons donc désormais d'utiliser une connexion unique pendant toute la durée du contexte d'exécution. Cela devrait inclure une certaine gestion des erreurs et un back-off-and-retry code standard pour gérer les fermetures inattendues de connexions.

Gestion des WebSocket connexions G705 dans les fonctions AWS Lambda

Si vous utilisez une variante du langage Gremlin pour interroger Neptune, le pilote se connecte à la base de données par le biais d'une connexion. WebSocket WebSockets sont conçus pour prendre en charge des scénarios de connexion client-serveur de longue durée. AWS Lambda, d'autre part, est conçu pour soutenir les exécutions apatrides et de courte durée. Cette différence de philosophie de conception peut entraîner des problèmes inattendus lors de l'utilisation de Lambda pour interroger Neptune.

Une AWS Lambda fonction s'exécute dans un contexte d'exécution qui l'isole des autres fonctions. Le contexte d'exécution est créé la première fois que la fonction est invoquée et peut être réutilisé pour les invocations ultérieures de la même fonction.

Cependant, aucun contexte d'exécution n'est jamais utilisé pour gérer plusieurs invocations simultanées de la fonction. Si la fonction est invoquée simultanément par plusieurs clients, Lambda crée un contexte d'exécution supplémentaire pour chaque instance de cette fonction. Tous ces nouveaux contextes d'exécution peuvent à leur tour être réutilisés pour les invocation suivantes de la fonction.

À un moment donné, Lambda recycle les contextes d'exécution, en particulier s'ils sont inactifs depuis un certain temps. AWS Lambda expose le cycle de vie du contexte d'exécution, y compris les `Shutdown` phases `Invoke` et`Init`, via les extensions Lambda. À l'aide de ces extensions, vous pouvez écrire du code qui nettoie les ressources externes telles que les connexions aux bases de données lorsque le contexte d'exécution est recyclé.

Une bonne pratique courante consiste à ouvrir la connexion à la base de données en dehors de la fonction du gestionnaire Lambda afin qu'elle puisse être réutilisée à chaque appel du gestionnaire. Si la connexion à la base de données est interrompue à un moment donné, vous pouvez vous reconnecter depuis le gestionnaire. Cependant, cette approche présente un risque de fuite de connexion. Si une connexion inactive reste ouverte longtemps après la destruction d'un contexte d'exécution, les scénarios d'invocation Lambda intermittents ou en paquets peuvent progressivement provoquer des fuites de connexions et épuiser les ressources de la base de données.

Les limites et délais de connexion de Neptune ont changé avec les nouvelles versions du moteur. Auparavant, chaque instance prenait en charge jusqu'à 60 000 WebSocket connexions. Désormais, le nombre maximum de WebSocket connexions simultanées par instance Neptune varie en fonction du type d'instance.

De plus, à partir de la version 1.0.3.0 du moteur, Neptune a réduit le délai d'inactivité des connexions d'une heure à environ 20 minutes. Si un client ne ferme pas la connexion, celle-ci est automatiquement fermée après un délai d'inactivité de 20 à 25 minutes. AWS Lambda ne documente pas les durées de vie des contextes d'exécution, mais les expériences montrent que le nouveau délai de connexion Neptune correspond bien aux délais d'expiration des contextes d'exécution Lambda inactifs. Au moment où un contexte d'exécution inactif est recyclé, il est très probable que sa connexion ait déjà été fermée par Neptune ou qu'elle le soit peu après.

Recommandations d'utilisation AWS Lambda avec Amazon Neptune G705

Nous recommandons désormais d'utiliser une seule source de connexion et de traversée de graphes pendant toute la durée de vie d'un contexte d'exécution Lambda, plutôt qu'une pour chaque

invocation de fonction (chaque invocation de fonction ne gère qu'une seule demande client). Comme les demandes simultanées des clients sont traitées par différentes instances de fonction exécutées dans des contextes d'exécution distincts, il n'est pas nécessaire de gérer un pool de connexions pour traiter les demandes simultanées au sein d'une instance de fonction. Si le pilote Gremlin que vous utilisez possède un pool de connexions, configurez-le pour n'utiliser qu'une seule connexion.

Pour gérer les échecs de connexion, utilisez une logique de nouvelle tentative pour chaque requête. Même si l'objectif est de maintenir une connexion unique pendant toute la durée de vie d'un contexte d'exécution, des événements réseau inattendus peuvent entraîner une interruption abrupte de cette connexion. Ces échecs de connexion se traduisent par des erreurs différentes selon le pilote que vous utilisez. Vous devez coder la fonction Lambda pour gérer ces problèmes de connexion et tenter une reconnexion si nécessaire.

Certains pilotes Gremlin gèrent automatiquement les reconnexions. Le pilote Java, par exemple, tente automatiquement de rétablir la connectivité à Neptune pour le compte du code client. Avec ce pilote, le code de la fonction n'a qu'à exécuter un backoff et une nouvelle tentative de la requête. Les pilotes Python JavaScript et Python, en revanche, n'implémentent aucune logique de reconnexion automatique. Ainsi, avec ces pilotes, votre code de fonction doit essayer de se reconnecter après avoir reculé, et ne réessayer la requête qu'une fois la connexion rétablie.

Les exemples de code présentés ici incluent la logique de reconnexion au lieu de supposer que le client s'en occupe.

Recommandations pour l'utilisation des demandes d'écriture Gremlin dans Lambda

Si votre fonction Lambda modifie les données du graphe, envisagez d'adopter une back-off-and-retry stratégie pour gérer les exceptions suivantes. Pour obtenir des conseils détaillés sur le développement d'une stratégie de réessai pratique, voirthe section called “Exceptions et nouvelles tentatives”.

- **`ConcurrentModificationException`** : la sémantique des transactions Neptune signifie que les demandes d'écriture échouent parfois avec une exception `ConcurrentModificationException`. Dans ces situations, essayez un mécanisme de back-off-based nouvelle tentative exponentielle.
- **`ReadOnlyViolationException`** : comme la topologie du cluster peut changer à tout moment en raison d'événements planifiés ou imprévus, les responsabilités d'écriture peuvent être transférées d'une instance du cluster à une autre. Si le code de la fonction tente d'envoyer une demande

d'écriture à une instance qui n'est plus l'instance principale (d'enregistreur), cette demande échouera avec une exception `ReadOnlyViolationException`. Dans ce cas, fermez la connexion existante, reconnectez-vous au point de terminaison du cluster, puis retentez la demande.

De plus, si vous utilisez une back-off-and-retry stratégie pour gérer les problèmes de demande d'écriture, envisagez d'implémenter des requêtes idempotentes pour les demandes de création et de mise à jour à l'aide de. the section called “Upserts efficaces à partir de la version 3.6.x”

Recommandations pour l'utilisation des demandes de lecture Gremlin dans Lambda

Si votre cluster possède un ou plusieurs réplicas en lecture, il est conseillé d'équilibrer les demandes de lecture entre ces réplicas. L'une des options consiste à utiliser le point de terminaison du lecteur. Le point de terminaison du lecteur équilibre les connexions entre les réplicas, même si la topologie du cluster change lorsque vous ajoutez ou supprimez des réplicas, ou lorsque vous promouvez un réplica pour en faire la nouvelle instance principale.

Cependant, l'utilisation du point de terminaison du lecteur peut entraîner une utilisation inégale des ressources du cluster dans certaines circonstances. Le point de terminaison de lecteur fonctionne en modifiant périodiquement l'hôte vers lequel pointe l'entrée DNS. Si un client ouvre de nombreuses connexions avant que l'entrée DNS ne change, toutes les demandes de connexion sont envoyées à une seule instance Neptune. Cela peut être le cas dans un scénario Lambda à haut débit dans lequel un grand nombre de demandes simultanées adressées à la fonction Lambda entraîne la création de plusieurs contextes d'exécution, chacun avec sa propre connexion. Si ces connexions sont toutes créées presque simultanément, elles sont susceptibles de toutes pointer vers le même réplica du cluster et de continuer à pointer vers lui jusqu'à ce que les contextes d'exécution soient recyclés.

L'un des moyens de répartir les demandes entre les instances consiste à configurer la fonction Lambda pour qu'elle se connecte à un point de terminaison d'instance, choisi au hasard dans une liste de points de terminaison d'instance de réplica, plutôt qu'au point de terminaison du lecteur. L'inconvénient de cette approche est qu'elle nécessite que le code Lambda gère les modifications de la topologie du cluster en surveillant le cluster et en mettant à jour la liste des points de terminaison chaque fois que l'appartenance au cluster change.

Si vous écrivez une fonction Lambda Java qui doit équilibrer les demandes de lecture entre les instances de votre cluster, vous pouvez utiliser le client Gremlin pour Amazon Neptune, un client

Java Gremlin qui connaît la topologie de votre cluster et qui répartit équitablement les connexions et les demandes entre un ensemble d'instances d'un cluster Neptune. Ce billet de blog inclut un exemple de fonction Lambda Java qui utilise le client Gremlin pour Amazon Neptune.

Facteurs susceptibles de ralentir les démarrages à froid des fonctions Lambda Neptune Gremlin

La première fois qu'une AWS Lambda fonction est invoquée, on parle de démarrage à froid. Plusieurs facteurs peuvent augmenter la latence d'un démarrage à froid :

- Assurez-vous d'affecter suffisamment de mémoire à la fonction Lambda. — La compilation pendant un démarrage à froid peut être nettement plus lente pour une fonction Lambda qu'elle ne le serait sur EC2, car les cycles du processeur sont AWS Lambda répartis de manière linéaire en fonction de la mémoire que vous attribuez à la fonction. Avec 1 769 Mo de mémoire, une fonction reçoit l'équivalent d'un vCPU complet (une seconde vCPU de crédits par seconde). L'impact de l'allocation d'une quantité de mémoire insuffisante pour recevoir les cycles de CPU est particulièrement prononcé pour les fonctions Lambda volumineuses écrites en Java.
- Sachez que l'activation de l'authentification de base de données IAM peut ralentir un démarrage à froid. L'authentification de base de données Gestion des identités et des accès AWS (IAM) peut également ralentir les démarrages à froid, en particulier si la fonction Lambda doit générer une nouvelle clé de signature. Cette latence n'affecte que le démarrage à froid et pas les demandes suivantes, car une fois que l'authentification de base de données IAM a établi les informations d'identification de connexion, Neptune ne fait que valider périodiquement qu'elles sont toujours valides.

Utilisation CloudFormation pour créer une fonction Lambda à utiliser dans Neptune

Vous pouvez utiliser un CloudFormation modèle pour créer une AWS Lambda fonction qui peut accéder à Neptune.

1. Pour lancer la pile de fonctions Lambda sur la CloudFormation console, cliquez sur l'un des boutons Launch Stack dans le tableau suivant.

Région	Afficher	Afficher dans Designer	Lancer
USA Est (Virginie du Nord)	Afficher	Afficher dans Designer	
USA Est (Ohio)	Afficher	Afficher dans Designer	
USA Ouest (Californie du Nord)	Afficher	Afficher dans Designer	
USA Ouest (Oregon)	Afficher	Afficher dans Designer	
Canada (Centre)	Afficher	Afficher dans Designer	
Amérique du Sud (São Paulo)	Afficher	Afficher dans Designer	
Europe (Stockholm)	Afficher	Afficher dans Designer	
Europe (Irlande)	Afficher	Afficher dans Designer	
Europe (Londres)	Afficher	Afficher dans Designer	
Europe (Paris)	Afficher	Afficher dans Designer	
Europe (Francfort)	Afficher	Afficher dans Designer	
Middle East (Bahrain)	Afficher	Afficher dans Designer	

Région	Afficher	Afficher dans Designer	Lancer
Moyen-Orient (EAU)	Afficher	Afficher dans Designer	Launch Stack
Israël (Tel Aviv)	Afficher	Afficher dans Designer	Launch Stack
Afrique (Le Cap)	Afficher	Afficher dans Designer	Launch Stack
Asie-Pacifique (Hong Kong)	Afficher	Afficher dans Designer	Launch Stack
Asie-Pacifique (Tokyo)	Afficher	Afficher dans Designer	Launch Stack
Asie-Pacifique (Séoul)	Afficher	Afficher dans Designer	Launch Stack
Asie-Pacifique (Singapour)	Afficher	Afficher dans Designer	Launch Stack
Asie-Pacifique (Sydney)	Afficher	Afficher dans Designer	Launch Stack
Asie-Pacifique (Mumbai)	Afficher	Afficher dans Designer	Launch Stack
Chine (Pékin)	Afficher	Afficher dans Designer	Launch Stack
Chine (Ningxia)	Afficher	Afficher dans Designer	Launch Stack
AWS GovCloud (US-West)	Afficher	Afficher dans Designer	Launch Stack

Région	Afficher	Afficher dans Designer	Lancer
AWS GovCloud (US-East)	Afficher	Afficher dans Designer	

2. Sur la page Select Template, choisissez Next.
3. Sur la page Specify Details (Spécifier les détails), définissez les options suivantes :

   a. Choisissez l'exécution Lambda qui correspond au langage que vous voulez utiliser dans votre fonction Lambda. Ces CloudFormation modèles sont actuellement compatibles avec les langues suivantes :

      - Python 3.9 (mappé avec `python39` dans l'URL Amazon S3)
      - NodeJS 18 (mappé avec `nodejs18x` dans l'URL Amazon S3)
      - Python 2.5 (mappé avec `ruby25` dans l'URL Amazon S3)

   b. Fournissez le point de terminaison et le numéro de port appropriés du cluster Neptune.

   c. Fournissez le groupe de sécurité Neptune approprié.

   d. Fournissez les paramètres de sous-réseau Neptune appropriés.
4. Choisissez Suivant.
5. Dans la page Options, choisissez Suivant.
6. Sur la page Vérification, cochez la première case pour accepter que CloudFormation créera des ressources IAM.

   Ensuite, choisissez Créer.

Si vous avez besoin d'apporter vos propres modifications à l'exécution Lambda, vous pouvez en télécharger une générique à partir d'un emplacement Amazon S3 dans votre région :

```
https://s3.Amazon region.amazonaws.com/aws-neptune-customer-samples-Amazon region/
lambda/runtime-language/lambda_function.zip.
```

Par exemple :

```
https://s3.us-west-2.amazonaws.com/aws-neptune-customer-samples-us-west-2/lambda/
python36/lambda_function.zip
```

AWS Lambda exemples de fonctions pour Amazon Neptune

Les exemples de AWS Lambda fonctions suivants, écrits en Java JavaScript et Python, illustrent l'insertion d'un seul sommet avec un ID généré aléatoirement à l'aide de l'`mergeV()`étape (voir). the section called “Upserts efficaces à partir de la version 3.6.x”

Une grande partie du code de chaque fonction est du code standard, chargé de gérer les connexions et de retenter les connexions et les requêtes en cas d'erreur. La logique d'application réelle et la requête Gremlin sont implémentées dans les méthodes `doQuery()` et `query()` respectivement. Si vous utilisez ces exemples comme base pour vos propres fonctions Lambda, vous pouvez vous concentrer sur la modification de `doQuery()` et `query()`.

Les fonctions sont configurées pour retenter les requêtes qui ont échoué cinq fois, en attendant une seconde entre les nouvelles tentatives. Dans une application de production, envisagez d'utiliser un recul exponentiel avec gigue plutôt qu'un intervalle fixe. Pour des conseils détaillés, voirthe section called “Exceptions et nouvelles tentatives”.

Les fonctions nécessitent la présence de valeurs dans les variables d'environnement Lambda suivantes :

- **`NEPTUNE_ENDPOINT`** : votre point de terminaison de cluster de bases de données Neptune. Pour Python, il devrait s'agir de `neptuneEndpoint`.
- **`NEPTUNE_PORT`** : port Neptune. Pour Python, il devrait s'agir de `neptunePort`.
- **`USE_IAM`** — (`true`ou`false`) Si l'authentification de base de données Gestion des identités et des accès AWS (IAM) est activée dans votre base de données, définissez la variable d'`USE_IAM``true`environnement sur. De cette manière, la fonction Lambda signera les demandes de connexion à Neptune avec Sigv4. Pour de telles demandes d'authentification de base de données IAM, assurez-vous que le rôle d'exécution de la fonction Lambda est associé à une politique IAM appropriée qui permet à la fonction de se connecter à votre cluster de bases de données Neptune (voir Types de politique IAM).

Exemple de fonction Lambda Java pour Amazon Neptune

Voici quelques points à garder à l'esprit à propos des AWS Lambda fonctions Java :

- Le pilote Java gère son propre pool de connexions, dont vous n'avez pas besoin. Configurez donc l'objet `Cluster` avec `minConnectionPoolSize(1)` et `maxConnectionPoolSize(1)`.

- La construction de l'objet `Cluster` peut être lente, car elle entraîne la création d'un ou de plusieurs sérialiseurs (Gyro par défaut, plus un autre si vous l'avez configuré pour des formats de sortie supplémentaires tels que `binary`). Leur instanciation peut prendre un certain temps.
- Le pool de connexions est initialisé lors de la première demande. À ce stade, le pilote configure la pile `Netty`, alloue des tampons d'octets et crée une clé de signature si vous utilisez l'authentification de base de données IAM. Tout cela peut augmenter la latence du démarrage à froid.
- Le pool de connexions du pilote Java surveille la disponibilité des hôtes du serveur et tente automatiquement de se reconnecter en cas d'échec de connexion. Il lance une tâche en arrière-plan pour tenter de rétablir la connexion. Utilisez `reconnectInterval( )` pour configurer l'intervalle entre les tentatives de reconnexion. Pendant que le pilote tente de se reconnecter, la fonction Lambda peut simplement retenter la requête.

  Si l'intervalle entre les nouvelles tentatives est inférieur à l'intervalle entre les tentatives de reconnexion, les nouvelles tentatives en cas d'échec de connexion échoueront à nouveau, car l'hôte sera considéré comme indisponible. Cela ne s'applique pas aux nouvelles tentatives pour une exception `ConcurrentModificationException`.
- Utilisez Java 8 plutôt que Java 11. Les optimisations `Netty` ne sont pas activées par défaut dans Java 11.
- Cet exemple utilise Retry4j pour les nouvelles tentatives.
- Pour utiliser le pilote de signature `Sigv4` dans la fonction Lambda Java, consultez les exigences de dépendance dans Connexion aux bases de données Amazon Neptune à l'aide d'IAM avec Gremlin Java.

> **Warning**
>
> Le `CallExecutor` de Retry4j n'est peut-être pas thread-safe. Pensez à faire en sorte que chaque thread utilise sa propre instance `CallExecutor`.

> **Note**
>
> L'exemple suivant a été mis à jour pour inclure l'utilisation de RequestInterceptor (). Cela a été ajouté dans la version TinkerPop 3.6.6. Avant la TinkerPop version 3.6.6, l'exemple de code utilisait HandshakeInterceptor (), qui était obsolète avec cette version.

```
package com.amazonaws.examples.social;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestStreamHandler;
import com.evanlennick.retry4j.CallExecutor;
import com.evanlennick.retry4j.CallExecutorBuilder;
import com.evanlennick.retry4j.Status;
import com.evanlennick.retry4j.config.RetryConfig;
import com.evanlennick.retry4j.config.RetryConfigBuilder;
import org.apache.tinkerpop.gremlin.driver.Cluster;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer;
import org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteConnection;
import org.apache.tinkerpop.gremlin.driver.ser.Serializers;
import org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource;
import org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversalSource;
import org.apache.tinkerpop.gremlin.process.traversal.Merge;
import org.apache.tinkerpop.gremlin.structure.T;

import java.io.*;
import java.time.temporal.ChronoUnit;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;
import java.util.concurrent.Callable;
import java.util.function.Function;

import static java.nio.charset.StandardCharsets.UTF_8;

public class MyHandler implements RequestStreamHandler {

  private final GraphTraversalSource g;
  private final CallExecutor<Object> executor;
  private final Random idGenerator = new Random();

  public MyHandler() {

    this.g = AnonymousTraversalSource
            .traversal()
            .withRemote(DriverRemoteConnection.using(createCluster()));


    this.executor = new CallExecutorBuilder<Object>()
```

```
            .config(createRetryConfig())
            .build();

    }

    @Override
    public void handleRequest(InputStream input,
                              OutputStream output,
                              Context context) throws IOException {

      doQuery(input, output);
    }

    private void doQuery(InputStream input, OutputStream output) throws IOException {
      try {

        Map<String, Object> args = new HashMap<>();
        args.put("id", idGenerator.nextInt());

        String result = query(args);

        try (Writer writer = new BufferedWriter(new OutputStreamWriter(output, UTF_8))) {
            writer.write(result);
        }

      } finally {
          input.close();
          output.close();
      }
    }

    private String query(Map<String, Object> args) {
      int id = (int) args.get("id");

      @SuppressWarnings("unchecked")
      Callable<Object> query = () -> g.mergeV(Map.of(T.id, id))
        .option(Merge.onCreate, Map.of(T.label, "Person"))
        .id().next();

      Status<Object> status = executor.execute(query);

      return status.getResult().toString();
    }
```

```
 private Cluster createCluster() {
   Cluster.Builder builder = Cluster.build()

.addContactPoint(System.getenv("NEPTUNE_ENDPOINT"))

.port(Integer.parseInt(System.getenv("NEPTUNE_PORT")))
                                    .enableSsl(true)
                                    .minConnectionPoolSize(1)
                                    .maxConnectionPoolSize(1)
                                    .serializer(Serializers.GRAPHBINARY_V1D0)
                                    .reconnectInterval(2000);

 if (Boolean.parseBoolean(getOptionalEnv("USE_IAM", "true"))) {
   // The following example uses requestInterceptor(), which was introduced
   // in TinkerPop 3.6.6. If you are using a TinkerPop version earlier than
   // 3.6.6 (but 3.5.5 or higher), use handshakeInterceptor() instead.
   builder.requestInterceptor( r ->
     {
       NeptuneNettyHttpSigV4Signer sigV4Signer = new
NeptuneNettyHttpSigV4Signer(region, DefaultCredentialsProvider.create());
       sigV4Signer.signRequest(r);
       return r;
     }
   )

   return builder.create();
 }

 // NOTE: This example uses a fixed backoff for simplicity.
 // In a production application consider use of exponential backoff with jitter.
 private RetryConfig createRetryConfig() {
   return new RetryConfigBuilder().retryOnCustomExceptionLogic(retryLogic())
                                   .withDelayBetweenTries(1000, ChronoUnit.MILLIS)
                                   .withMaxNumberOfTries(5)
                                   .withFixedBackoff()
                                   .build();
 }

 private Function<Exception, Boolean> retryLogic() {
   return e -> {
     StringWriter stringWriter = new StringWriter();
     e.printStackTrace(new PrintWriter(stringWriter));
     String message = stringWriter.toString();
```

```
      // Check for connection issues
      if ( message.contains("Timed out while waiting for an available host") ||
           message.contains("Timed-out waiting for connection on Host") ||
           message.contains("Connection to server is no longer active") ||
           message.contains("Connection reset by peer") ||
           message.contains("SSLEngine closed already") ||
           message.contains("Pool is shutdown") ||
           message.contains("ExtendedClosedChannelException") ||
           message.contains("Broken pipe")) {
        return true;
      }

      // Concurrent writes can sometimes trigger a ConcurrentModificationException.
      // In these circumstances you may want to backoff and retry.
      if (message.contains("ConcurrentModificationException")) {
          return true;
      }

      // If the primary fails over to a new instance, existing connections to the old
 primary will
      // throw a ReadOnlyViolationException. You may want to back and retry.
      if (message.contains("ReadOnlyViolationException")) {
          return true;
      }

      return false;
    };
  }

  private String getOptionalEnv(String name, String defaultValue) {
    String value = System.getenv(name);
    if (value != null && value.length() > 0) {
      return value;
    } else {
      return defaultValue;
    }
  }
}
```

Si vous souhaitez inclure une logique de reconnexion dans votre fonction, consultez Exemple de reconnexion Java.

JavaScript Exemple de fonction Lambda pour Amazon Neptune

Remarques à propos de cet exemple

- Le JavaScript pilote ne gère pas de pool de connexions. Il ouvre toujours une seule connexion.
- L'exemple de fonction utilise les utilitaires de signature Sigv4 de gremlin-aws-sigv4 pour signer des demandes vers une base de données activée pour l'authentification IAM.
- Il utilise la fonction retry () du module utilitaire asynchrone open source pour gérer les tentatives. backoff-and-retry
- Les étapes du terminal Gremlin renvoient un JavaScript `promise` (voir la TinkerPop documentation). Pour `next()`, il s'agit d'un tuple `{value, done}`.
- Les erreurs de connexion sont signalées dans le gestionnaire et traitées selon une certaine backoff-and-retry logique conformément aux recommandations décrites ici, à une exception près. Il existe un type de problème de connexion que le pilote ne considère pas comme une exception et qui ne peut donc pas être résolu par cette backoff-and-retry logique.

  Le problème est que si une connexion est fermée après qu'un pilote a envoyé une demande, mais avant que le pilote ne reçoive une réponse, la requête semble s'être terminée, mais renvoie une valeur nulle. En ce qui concerne le client de la fonction Lambda, celle-ci semble s'être terminée correctement, mais avec une réponse vide.

  L'impact de ce problème dépend de la manière dont votre application traite une réponse vide. Certaines applications peuvent traiter une réponse vide provenant d'une demande de lecture comme une erreur, tandis que d'autres peuvent la traiter à tort comme un résultat vide.

  Les demandes d'écriture qui rencontrent ce problème de connexion renvoient également une réponse vide. Une invocation réussie avec une réponse vide signifie-t-elle un succès ou un échec ? Si le client invoquant une fonction d'écriture considère simplement l'invocation réussie de la fonction comme signifiant que l'écriture dans la base de données a été validée, au lieu d'inspecter le corps de la réponse, le système peut sembler perdre des données.

  Ce problème est dû à la façon dont le pilote traite les événements émis par le socket sous-jacent. Lorsque le socket réseau sous-jacent est fermé avec une `ECONNRESET` erreur, le socket WebSocket utilisé par le pilote est fermé et émet un `'ws close'` événement. Cependant, rien dans le pilote ne permet de gérer cet événement d'une manière qui pourrait être utilisée pour déclencher une exception. Par conséquent, la requête disparaît tout simplement.

Pour contourner ce problème, l'exemple de fonction lambda ci-dessous ajoute un gestionnaire d'événements `'ws close'` qui envoie une exception au pilote lors de la création d'une connexion à distance. Cette exception n'est toutefois pas levée le long du chemin requête-réponse de la requête Gremlin et ne peut donc pas être utilisée pour déclencher une quelconque backoff-and-retry logique au sein de la fonction lambda elle-même. Au lieu de cela, l'exception émise par le gestionnaire d'événements `'ws close'` entraîne une exception non gérée qui provoque l'échec de l'invocation Lambda. Cela permet au client qui invoque la fonction de gérer l'erreur et de retenter l'invocation Lambda le cas échéant.

Nous vous recommandons d'implémenter une backoff-and-retry logique dans la fonction lambda elle-même afin de protéger vos clients des problèmes de connexion intermittents. Cependant, pour contourner le problème ci-dessus, le client doit également implémenter une logique de nouvelle tentative afin de gérer les échecs résultant de ce problème de connexion particulier.

Code Javascript

```
const gremlin = require('gremlin');
const async = require('async');
const {getUrlAndHeaders} = require('gremlin-aws-sigv4/lib/utils');

const traversal = gremlin.process.AnonymousTraversalSource.traversal;
const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;
const t = gremlin.process.t;
const Merge = gremlin.process.merge;

let conn = null;
let g = null;

async function query(context) {

  const id = context.id;

  return g.mergeV({[t.id]: id})
    .option(Merge.onCreate, {[t.label]: 'User'})
    .id().next();
}

async function doQuery() {
  const id = Math.floor(Math.random() * 10000).toString();
```

```
  let result = await query({id: id});
  return result['value'];
}

exports.handler = async (event, context) => {

  const getConnectionDetails = () => {
    if (process.env['USE_IAM'] == 'true'){
       return getUrlAndHeaders(
         process.env['NEPTUNE_ENDPOINT'],
         process.env['NEPTUNE_PORT'],
         {},
         '/gremlin',
         'wss');
    } else {
      const database_url = 'wss://' + process.env['NEPTUNE_ENDPOINT'] + ':' +
 process.env['NEPTUNE_PORT'] + '/gremlin';
      return { url: database_url, headers: {}};
    }
  };


  const createRemoteConnection = () => {
    const { url, headers } = getConnectionDetails();

    const c = new DriverRemoteConnection(
      url,
      {
        headers: headers
      });

     c._client._connection.on('close', (code, message) => {
         console.info(`close - ${code} ${message}`);
         if (code == 1006){
           console.error('Connection closed prematurely');
           throw new Error('Connection closed prematurely');
         }
       });

     return c;
  };

  const createGraphTraversalSource = (conn) => {
    return traversal().withRemote(conn);
```

```
};

if (conn == null){
  console.info("Initializing connection")
  conn = createRemoteConnection();
  g = createGraphTraversalSource(conn);
}

// NOTE: This example uses a fixed interval for simplicity.
// In a production application consider use of exponential backoff with jitter.
return async.retry(
  {
    times: 5,
    interval: 1000,
    errorFilter: function (err) {

      // Add filters here to determine whether error can be retried
      console.warn('Determining whether retriable error: ' + err.message);

      // Check for connection issues
      if (err.message.startsWith('WebSocket is not open')){
        console.warn('Reopening connection');
        conn.close();
        conn = createRemoteConnection();
        g = createGraphTraversalSource(conn);
        return true;
      }

      // Check for ConcurrentModificationException
      if (err.message.includes('ConcurrentModificationException')){
        console.warn('Retrying query because of ConcurrentModificationException');
        return true;
      }

      // Check for ReadOnlyViolationException
      if (err.message.includes('ReadOnlyViolationException')){
        console.warn('Retrying query because of ReadOnlyViolationException');
        return true;
      }

      return false;
    }

  },
```

```
    doQuery);
};
```

Exemple de fonction Lambda Python pour Amazon Neptune

Voici quelques points à noter à propos de l'exemple de fonction AWS Lambda Python suivant :

- Il utilise le module de backoff.
- Il configure `pool_size=1` pour éviter de créer un pool de connexions inutile.

```
import os, sys, backoff, math
from random import randint
from gremlin_python import statics
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection
from gremlin_python.driver.protocol import GremlinServerError
from gremlin_python.driver import serializer
from gremlin_python.process.anonymous_traversal import traversal
from gremlin_python.process.graph_traversal import __
from gremlin_python.process.strategies import *
from gremlin_python.process.traversal import T
from aiohttp.client_exceptions import ClientConnectorError
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import ReadOnlyCredentials
from types import SimpleNamespace

import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)


reconnectable_err_msgs = [
    'ReadOnlyViolationException',
    'Server disconnected',
    'Connection refused',
    'Connection was already closed',
    'Connection was closed by server',
    'Failed to connect to server: HTTP Error code 403 - Forbidden'
]

retriable_err_msgs = ['ConcurrentModificationException'] + reconnectable_err_msgs
```

```
network_errors = [OSError, ClientConnectorError]

retriable_errors = [GremlinServerError, RuntimeError, Exception] + network_errors

def prepare_iamdb_request(database_url):

    service = 'neptune-db'
    method = 'GET'

    access_key = os.environ['AWS_ACCESS_KEY_ID']
    secret_key = os.environ['AWS_SECRET_ACCESS_KEY']
    region = os.environ['AWS_REGION']
    session_token = os.environ['AWS_SESSION_TOKEN']


    creds = SimpleNamespace(
        access_key=access_key, secret_key=secret_key, token=session_token,
 region=region,
    )

    request = AWSRequest(method=method, url=database_url, data=None)
    SigV4Auth(creds, service, region).add_auth(request)

    return database_url, request.headers.items()

def is_retriable_error(e):

    is_retriable = False
    err_msg = str(e)

    if isinstance(e, tuple(network_errors)):
        is_retriable = True
    else:
        is_retriable = any(retriable_err_msg in err_msg for retriable_err_msg in
 retriable_err_msgs)

    logger.error('error: [{}] {}'.format(type(e), err_msg))
    logger.info('is_retriable: {}'.format(is_retriable))

    return is_retriable

def is_non_retriable_error(e):
    return not is_retriable_error(e)
```

```
def reset_connection_if_connection_issue(params):

    is_reconnectable = False

    e = sys.exc_info()[1]
    err_msg = str(e)

    if isinstance(e, tuple(network_errors)):
        is_reconnectable = True
    else:
        is_reconnectable = any(reconnectable_err_msg in err_msg for
 reconnectable_err_msg in reconnectable_err_msgs)

    logger.info('is_reconnectable: {}'.format(is_reconnectable))

    if is_reconnectable:
        global conn
        global g
        conn.close()
        conn = create_remote_connection()
        g = create_graph_traversal_source(conn)

@backoff.on_exception(backoff.constant,
                      tuple(retriable_errors),
                      max_tries=5,
                      jitter=None,
                      giveup=is_non_retriable_error,
                      on_backoff=reset_connection_if_connection_issue,
                      interval=1)
def query(**kwargs):

    id = kwargs['id']

    return (g.V().hasLabel('column').has('column_name',
 'amhstr_ag_type').in_('hascolumn').dedup().valueMap().limit(10).toList())

def doQuery(event):
    return query(id=str(randint(0, 10000)))

def lambda_handler(event, context):
    result = doQuery(event)
    logger.info('result - {}'.format(result))
    return result
```

```
def create_graph_traversal_source(conn):
    return traversal().withRemote(conn)

def create_remote_connection():
    logger.info('Creating remote connection')

    (database_url, headers) = connection_info()

    # Convert headers to a dictionary if it's not already
    headers_dict = dict(headers) if isinstance(headers, list) else headers

    print(headers)
    return DriverRemoteConnection(
        database_url,
        'g',
        pool_size=1,
        headers=headers_dict)


def connection_info():

    database_url = 'wss://{}:{}/gremlin'.format(os.environ['neptuneEndpoint'],
 os.environ['neptunePort'])

    if 'USE_IAM' in os.environ and os.environ['USE_IAM'] == 'true':
        return prepare_iamdb_request(database_url)
    else:
        return (database_url, {})

conn = create_remote_connection()
g = create_graph_traversal_source(conn)
```

Voici des exemples de résultats, illustrant des périodes alternées de charge lourde et légère :

GremlinWebSocketOpenConnections
Count
261
131
0
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
neptunedbinstance-l22qsahgvgvl
replica-python-lambdas-neptunebasestack-5rzhq9jvp2ri
GremlinRequestsPreSecond
Count/Second
861
430
0
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
neptunedbinstance-l22qsahgvgvl
replica-python-lambdas-neptunebasestack-5rzhq9jvp2ri
Lambda-ConcurrentExecutions
Count
86
43.5
1
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
social-b6501310_add-follows-relationship social-b6501310_add-follows-relationship
social-b6501310_create-user social-b6501310_create-user
social-b6501310_get-recent-follows social-b6501310_get-recent-follows
social-b6501310_get-user-details social-b6501310_get-user-details
Lambda-P99-Latency
Milliseconds
2.10k
1.05k
5.46
06:05 06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
social-b6501310_add-follows-relationship
social-b6501310_create-user
social-b6501310_get-recent-follows
social-b6501310_get-user-details
social-b6501310_recommend-new-follows
Workload
Various units
Count
2.17k
1.09k
9e-3
147k
73.7k
2
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
P99Latency
AverageThroughput
Succeeded
Errors
Count
1
0.5
0
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
social-b6501310_add-follows-relationship
social-b6501310_create-user
social-b6501310_get-recent-follows
social-b6501310_get-user-details
social-b6501310_recommend-new-follows

Amazon Neptune ML pour le machine learning sur les graphes

Les grands jeux de données connectés contiennent souvent des informations précieuses qu'il peut être difficile d'extraire sur la seule base de l'intuition humaine. Les techniques de machine learning (ML) peuvent aider à trouver des corrélations cachées dans des graphes contenant des milliards de relations. Ces corrélations peuvent être utiles pour recommander des produits, prédire la solvabilité, identifier des fraudes et bien d'autres choses encore.

La fonctionnalité Neptune ML permet de créer et d'entraîner des modèles de machine learning utiles sur de grands graphes en quelques heures au lieu de plusieurs semaines. Pour ce faire, Neptune ML utilise la technologie GNN (Graph Neural Network) développée par Amazon SageMaker AI et la Deep Graph Library (DGL) (qui est open source). Les réseaux de neurones en graphes sont un domaine émergent de l'intelligence artificielle (consultez, par exemple, A Comprehensive Survey on Graph Neural Networks). Pour un didacticiel pratique sur l'utilisation GNNs de DGL, voir Apprendre les réseaux neuronaux à graphes avec Deep Graph Library.

> **Note**
>
> Les sommets d'un graphe sont identifiés dans les modèles Neptune ML en tant que « nœuds ». Par exemple, la classification de sommet utilise un modèle de machine learning de classification de nœud, tandis que la régression de sommet utilise un modèle de régression de nœud.

Ce que Neptune ML peut faire

Neptune prend en charge à la fois l'inférence transductive, qui renvoie des prédictions précalculées au moment de l'entraînement, en fonction de vos données de graphe à ce moment-là, et l'inférence inductive, qui renvoie et applique le traitement des données et l'évaluation du modèle en temps réel, sur la base des données actuelles. Consultez Différence entre inférence inductive et inférence transductive.

Neptune ML peut entraîner des modèles de machine learning pour prendre en charge cinq catégories d'inférence différentes :

Types de tâche d'inférence actuellement pris en charge par Neptune ML

- Classification de nœud : prédiction de la fonctionnalité catégorielle d'une propriété de sommet.

  Par exemple, pour le film Les Évadés, Neptune ML peut prédire sa propriété `genre` en tant que `story` à partir d'un ensemble de candidats `[story, crime, action, fantasy, drama, family, ...]`.

  Il existe deux types de tâches de classification de nœud :

  - Classification à classe unique : dans ce type de tâche, chaque nœud possède une seule fonctionnalité cible. Par exemple, la propriété `Place_of_birth` de `Alan Turing` a la valeur `UK`.
  - Classification multiclasse : dans ce type de tâche, chaque nœud peut avoir plus d'une seule fonctionnalité cible. Par exemple, la propriété `genre` du film Le Parrain a les valeurs `crime` et `story`.
- Régression de nœud : prédiction d'une propriété numérique d'un sommet.

  Par exemple, pour le film Avengers, le dernier affrontement, Neptune ML peut prédire que sa propriété `popularity` a une valeur de `5.0`.
- Classification d'arête : prédiction de la fonctionnalité catégorielle d'une propriété d'arête.

  Il existe deux types de tâches de classification d'arête :

  - Classification à classe unique : dans ce type de tâche, chaque arête possède une seule fonctionnalité cible. Par exemple, une arête d'évaluation entre un utilisateur et un film peut avoir la propriété `liked`, avec la valeur « Oui » ou « Non ».
  - Classification multiclasse : dans ce type de tâche, chaque arête peut avoir plus d'une seule fonctionnalité cible. Par exemple, une évaluation entre un utilisateur et un film peut avoir plusieurs valeurs de balise de propriété, telles que « Drôle », « Réconfortant », « Reposant », etc.
- Régression d'arête : prédiction d'une propriété numérique d'une arête.

  Par exemple, une arête d'évaluation entre un utilisateur et un film peut avoir la propriété numérique `score`, pour laquelle Neptune ML peut prédire une valeur à partir d'un utilisateur et d'un film.
- Prédiction de lien : prédiction des nœuds de destination les plus probables pour un nœud source et une arête sortante particuliers, ou des nœuds sources les plus probables pour un nœud de destination et une arête entrante donnés.

Par exemple, avec un graphe de connaissances médicament-maladie, avec `Aspirin` comme nœud source et `treats` comme arête sortante, Neptune ML peut prédire les nœuds de destination les plus pertinents `heart disease`, `fever`, etc.

Ou, avec le graphe de connaissances de Wikimedia, avec `President-of` comme arête ou relation et `United-States` comme nœud de destination, Neptune ML peut prédire les têtes les plus pertinentes `George Washington`, `Abraham Lincoln`, `Franklin D. Roosevelt`, etc.

> **Note**
>
> La classification de nœud et la classification d'arête prennent en charge uniquement des valeurs de chaîne. Cela signifie que des valeurs de propriété numériques telles que `0` ou `1` ne sont pas prises en charge, alors que les valeurs équivalentes de type string `"0"` et `"1"` le sont. De même, les valeurs de propriété booléenne `true` et `false` ne fonctionnent pas, mais `"true"` et `"false"` fonctionnent.

Avec Neptune ML, vous pouvez utiliser des modèles de machine learning séparés en deux catégories générales :

Types de modèles de machine learning actuellement pris en charge par Neptune ML

- Modèles de réseaux neuronaux à graphes (GNN) : ils incluent les réseaux convolutionnels à graphes relationnels (R -). GCNs Les modèles GNN fonctionnent pour les trois types de tâches ci-dessus.
- Modèles d'intégration de graphe de connaissances (KGE) : ils incluent les modèles `TransE`, `DistMult` et `RotatE`. Ils fonctionnent uniquement pour la prédiction de lien.

Modèles définis par l'utilisateur : Neptune ML vous permet également de fournir votre propre implémentation de modèle personnalisé pour tous les types de tâches répertoriés ci-dessus. Vous pouvez utiliser la boîte à outils Neptune ML pour développer et tester votre implémentation de modèle personnalisé basée sur Python avant d'utiliser l'API d'entraînement Neptune ML avec votre modèle. Consultez Modèles personnalisés dans Neptune ML pour plus de détails sur la manière de structurer et d'organiser votre implémentation afin qu'elle soit compatible avec l'infrastructure d'entraînement de Neptune ML.

Configuration de Neptune ML

Le moyen le plus simple de démarrer avec Neptune ML est d'utiliser le modèle de démarrage CloudFormation rapide. Ce modèle installe tous les composants nécessaires, y compris un nouveau cluster de bases de données Neptune, tous les rôles IAM nécessaires et un nouveau bloc-notes de graphe Neptune pour faciliter l'utilisation de Neptune ML.

Vous pouvez également installer Neptune ML manuellement, comme expliqué dans Configuration de Neptune ML sans utiliser le modèle de démarrage rapide CloudFormation.

Utilisation du Neptune ML AWS CloudFormation modèle pour démarrer rapidement dans un nouveau cluster de base de données

Le moyen le plus simple de démarrer avec Neptune ML est d'utiliser le modèle de démarrage CloudFormation rapide. Ce modèle installe tous les composants nécessaires, y compris un nouveau cluster de bases de données Neptune, tous les rôles IAM nécessaires et un nouveau bloc-notes de graphe Neptune pour faciliter l'utilisation de Neptune ML.

Pour créer la pile de démarrage rapide Neptune ML

1. Pour lancer la CloudFormation pile sur la CloudFormation console, cliquez sur l'un des boutons Launch Stack dans le tableau suivant :

Région	Afficher	Lancer
USA Est (Virginie du Nord)	Afficher	
USA Est (Ohio)	Afficher	
USA Ouest (Californie du Nord)	Afficher	
USA Ouest (Oregon)	Afficher	
Canada (Centre)	Afficher	
Amérique du Sud (São Paulo)	Afficher	
Europe (Stockholm)	Afficher	
Europe (Irlande)	Afficher	
Europe (Londres)	Afficher	

Région	Afficher	Lancer
Europe (Paris)	Afficher	
Europe (Francfort)	Afficher	
Middle East (Bahrain)	Afficher	
Moyen-Orient (EAU)	Afficher	
Israël (Tel Aviv)	Afficher	
Afrique (Le Cap)	Afficher	
Asie-Pacifique (Hong Kong)	Afficher	
Asie-Pacifique (Tokyo)	Afficher	
Asie-Pacifique (Séoul)	Afficher	
Asie-Pacifique (Singapour)	Afficher	
Asie-Pacifique (Sydney)	Afficher	
Asie-Pacifique (Mumbai)	Afficher	
Chine (Pékin)	Afficher	
Chine (Ningxia)	Afficher	
AWS GovCloud (US-West)	Afficher	

2. Sur la page Select Template, choisissez Next.
3. Sur la page Specify Details (Spécifier les détails), choisissez Next (Suivant).
4. Dans la page Options, choisissez Suivant.
5. Sur la page Vérifier, vous devez cocher deux cases :

   - Le premier reconnaît que cela AWS CloudFormation peut créer des ressources IAM avec des noms personnalisés.
   - La seconde reconnaît que cela AWS CloudFormation pourrait nécessiter la `CAPABILITY_AUTO_EXPAND` capacité de la nouvelle pile. `CAPABILITY_AUTO_EXPAND`permet explicitement d' CloudFormation étendre automatiquement les macros lors de la création de la pile, sans révision préalable.

     Les clients créent souvent un jeu de modifications à partir d'un modèle traité, de sorte que les modifications apportées par les macros puissent être révisées avant la création de la pile. Pour en savoir plus, consultez l'API CloudFormation CreateStack.

   Ensuite, choisissez Créer.

Le modèle de démarrage rapide crée et configure les éléments suivants :

- Un cluster de bases de données Neptune.
- Les rôles IAM nécessaires (et les attache).
- Le groupe de sécurité Amazon EC2 nécessaire.
- Les points de SageMaker terminaison VPC AI nécessaires.
- Un groupe de paramètres de cluster de bases de données pour Neptune ML.
- Les paramètres nécessaires dans ce groupe de paramètres.
- Un bloc-notes SageMaker basé sur l'IA avec des exemples de blocs-notes préremplis pour Neptune ML. Notez que toutes les tailles d'instance ne sont pas disponibles dans chaque région. Vous devez donc vous assurer que la taille d'instance de bloc-notes sélectionnée est prise en charge par votre région.
- Le Neptune-Export service.

Lorsque la pile de démarrage rapide est prête, accédez au bloc-notes SageMaker AI créé par le modèle et consultez les exemples préremplis. Ils vous aideront à télécharger des exemples de jeux de données à utiliser pour expérimenter les fonctionnalités de Neptune ML.

Ils peuvent également vous faire gagner beaucoup de temps dans le cadre de l'utilisation de Neptune ML. Découvrez par exemple la commande magique linéaire %neptune_ml et la commande magique cellulaire %%neptune_ml prises en charge par ces blocs-notes.

Vous pouvez également utiliser la AWS CLI commande suivante pour exécuter le modèle de démarrage rapide CloudFormation :

```
aws cloudformation create-stack \
  --stack-name neptune-ml-fullstack-$(date '+%Y-%m-%d-%H-%M') \
  --template-url https://aws-neptune-customer-samples.s3.amazonaws.com/v2/
cloudformation-templates/neptune-ml-nested-stack.json \
  --parameters ParameterKey=EnableIAMAuthOnExportAPI,ParameterValue=(true if you have
 IAM auth enabled, or false otherwise) \
               ParameterKey=Env,ParameterValue=test$(date '+%H%M')\
  --capabilities CAPABILITY_IAM \
  --region (the AWS region, like us-east-1) \
  --disable-rollback \
  --profile (optionally, a named CLI profile of yours)
```

Configuration de Neptune ML sans utiliser le modèle de démarrage rapide CloudFormation

Ce guide fournit des step-by-step instructions pour configurer Amazon Neptune ML sans utiliser le modèle de démarrage AWS CloudFormation rapide. Il suppose que vous disposez déjà d'un cluster de base de données Neptune fonctionnel et couvre la configuration nécessaire, y compris l'installation du service Neptune-Export, la création de rôles IAM personnalisés et la configuration de votre cluster de base de données pour activer Neptune ML. Le guide explique également comment créer deux points de terminaison SageMaker IA dans votre VPC Neptune pour permettre au moteur Neptune d'accéder à la gestion de l'IA nécessaire. SageMaker APIs En suivant ces instructions, vous pouvez configurer Neptune ML sur votre infrastructure Neptune existante sans vous fier au modèle. CloudFormation

Utilisation d'un cluster de bases de données Neptune fonctionnel

Si vous n'utilisez pas le modèle de CloudFormation démarrage rapide pour configurer Neptune ML, vous aurez besoin d'un cluster de base de données Neptune existant pour fonctionner. Si vous le souhaitez, vous pouvez en utiliser un que vous possédez déjà, ou en cloner un que vous utilisez déjà, ou encore en créer un nouveau (voir Créer un cluster Neptune).

Installation du service d'exportation Neptune

Si vous ne l'avez pas encore fait, installez le service d'exportation Neptune, comme expliqué dans Utilisation du Neptune-Export service pour exporter les données Neptune.

Ajoutez une règle entrante au groupe de sécurité `NeptuneExportSecurityGroup` créé par l'installation, avec les paramètres suivants :

- Type : `Custom TCP`
- Protocole : `TCP`
- Plage de ports : `80 - 443`
- Source : *`(Neptune DB cluster security group ID)`*

Création d'un rôle NeptuneLoadFrom S3 IAM personnalisé

Si vous ne l'avez pas encore fait, créez un rôle IAM `NeptuneLoadFromS3` personnalisé, comme expliqué dans Création d'une politique IAM pour accéder à Amazon S3.

Création d'un NeptuneSageMaker IAMRole rôle personnalisé

Utilisez la console IAM pour créer un `NeptuneSageMakerIAMRole` personnalisé, en utilisant la politique suivante :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:CreateNetworkInterfacePermission",
        "ec2:CreateVpcEndpoint",
        "ec2:DeleteNetworkInterface",
        "ec2:DeleteNetworkInterfacePermission",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
```

```
        "arn:aws:iam::*:role/*"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "sagemaker.amazonaws.com"
          ]
        }
      },
      "Effect": "Allow"
    },
    {
      "Action": [
        "kms:CreateGrant",
        "kms:Decrypt",
        "kms:GenerateDataKey*"
      ],
      "Resource": "arn:aws:kms:*:*:key/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/sagemaker/*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:AddTags",
        "sagemaker:CreateEndpoint",
        "sagemaker:CreateEndpointConfig",
        "sagemaker:CreateHyperParameterTuningJob",
        "sagemaker:CreateModel",
        "sagemaker:CreateProcessingJob",
        "sagemaker:CreateTrainingJob",
        "sagemaker:CreateTransformJob",
```

```
        "sagemaker:DeleteEndpoint",
        "sagemaker:DeleteEndpointConfig",
        "sagemaker:DeleteModel",
        "sagemaker:DescribeEndpoint",
        "sagemaker:DescribeEndpointConfig",
        "sagemaker:DescribeHyperParameterTuningJob",
        "sagemaker:DescribeModel",
        "sagemaker:DescribeProcessingJob",
        "sagemaker:DescribeTrainingJob",
        "sagemaker:DescribeTransformJob",
        "sagemaker:InvokeEndpoint",
        "sagemaker:ListTags",
        "sagemaker:ListTrainingJobsForHyperParameterTuningJob",
        "sagemaker:StopHyperParameterTuningJob",
        "sagemaker:StopProcessingJob",
        "sagemaker:StopTrainingJob",
        "sagemaker:StopTransformJob",
        "sagemaker:UpdateEndpoint",
        "sagemaker:UpdateEndpointWeightsAndCapacities"
      ],
      "Resource": [
        "arn:aws:sagemaker:*:*:*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:ListEndpointConfigs",
        "sagemaker:ListEndpoints",
        "sagemaker:ListHyperParameterTuningJobs",
        "sagemaker:ListModels",
        "sagemaker:ListProcessingJobs",
        "sagemaker:ListTrainingJobs",
        "sagemaker:ListTransformJobs"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:AbortMultipartUpload",
```

```
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Lors de la création de ce rôle, modifiez la relation d'approbation afin qu'elle se présente comme suit :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "ec2.amazonaws.com",
          "rds.amazonaws.com",
          "sagemaker.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Enfin, copiez l'ARN attribué à ce nouveau rôle `NeptuneSageMakerIAMRole`.

⚠ Important

- Veillez à ce que les autorisations Amazon S3 figurant dans `NeptuneSageMakerIAMRole` correspondent aux autorisations ci-dessus.
- L'ARN universel `arn:aws:s3:::*` est utilisé pour la ressource Amazon S3 dans la politique ci-dessus. Si, pour une raison ou une autre, l'ARN universel ne peut pas être

utilisé, `arn:aws:s3:::graphlytics*` et l'ARN de toute autre ressource Amazon S3 du client que les commandes NeptuneML utiliseront doivent être ajoutés à la section des ressources.

Configuration de votre cluster de bases de données de façon à activer Neptune ML

Pour configurer votre cluster de bases de données pour Neptune ML

1. Dans la console Neptune, accédez à Groupes de paramètres, puis au groupe de paramètres du cluster de bases de données associé au cluster de bases de données que vous allez utiliser. Définissez le paramètre `neptune_ml_iam_role` sur l'ARN attribué au rôle `NeptuneSageMakerIAMRole` que vous venez de créer.
2. Accédez à Bases de données, puis sélectionnez le cluster de bases de données que vous utiliserez pour Neptune ML. Sélectionnez Actions, puis Gérer les rôles IAM.
3. Sur la page Gérer les rôles IAM, sélectionnez Ajouter un rôle et ajoutez `NeptuneSageMakerIAMRole`. Ajoutez ensuite le rôle `NeptuneLoadFromS3`.
4. Redémarrez l'instance d'enregistreur de votre cluster de bases de données.

Créez deux points de terminaison SageMaker IA dans votre VPC Neptune

Enfin, pour donner au moteur Neptune l'accès à la gestion de l' SageMaker IA nécessaire APIs, vous devez créer deux points de terminaison d' SageMaker IA dans votre VPC Neptune, comme expliqué dans. Créez deux points de terminaison pour l' SageMaker IA dans votre VPC Neptune

Configuration manuelle d'un bloc-notes Neptune pour Neptune ML

Les blocs-notes Neptune SageMaker AI sont préchargés avec une variété d'exemples de blocs-notes pour Neptune ML. Vous pouvez prévisualiser ces exemples dans le référentiel open source graph-notebook GitHub .

Vous pouvez utiliser l'un des blocs-notes Neptune existants ou, si vous le souhaitez, créer le vôtre en suivant les instructions fournies dans Utilisation du workbench Neptune pour héberger des blocs-notes Neptune.

Vous pouvez également configurer un bloc-notes Neptune par défaut à utiliser avec Neptune ML en procédant comme suit :

Modification d'un bloc-notes pour Neptune ML

1. Ouvrez la console Amazon SageMaker AI à l'adresse https://console.aws.amazon.com/sagemaker/.
2. Dans le panneau de navigation de gauche, choisissez Bloc-notes, puis Instances de blocs-notes. Recherchez le nom du bloc-notes Neptune que vous souhaitez utiliser pour Neptune ML et sélectionnez-le pour accéder à sa page de détails.
3. Si l'instance de bloc-notes est en cours d'exécution, sélectionnez le bouton Arrêter en haut à droite de la page de détails du bloc-notes.
4. Dans Paramètres d'instances de blocs-notes, sous Configuration du cycle de vie, sélectionnez le lien pour ouvrir la page du cycle de vie du bloc-notes.
5. Sélectionnez Modifier en haut à droite, puis Continuer.
6. Dans l'onglet Démarrer un bloc-notes, modifiez le script pour inclure des commandes d'exportation supplémentaires et pour remplir les champs correspondant à votre rôle IAM Neptune ML et à l'URI du service d'exportation, similaires à ce qui suit selon votre shell :

```
echo "export NEPTUNE_ML_ROLE_ARN=(your Neptune ML IAM role ARN)" >> ~/.bashrc
echo "export NEPTUNE_EXPORT_API_URI=(your export service URI)" >> ~/.bashrc
```

7. Tâche de sélection Update (Mise à jour).
8. Revenez à la page d'instance de bloc-notes. Sous Autorisations et chiffrement, il existe un champ pour l'ARN du rôle IAM. Sélectionnez le lien dans ce champ pour accéder au rôle IAM avec lequel cette instance de bloc-notes sera exécutée.
9. Créez une nouvelle politique en ligne comme celle-ci :

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "arn:aws:cloudwatch:us-east-1:111122223333:*",
      "Sid": "AllowPutMetrics",
      "Effect": "Allow"
```

```
    },
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "arn:aws:logs:us-east-1:111122223333:*",
      "Sid": "AllowCreateLogs",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:Put*",
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "arn:aws:s3:::*",
      "Sid": "AllowS3Actions",
      "Effect": "Allow"
    },
    {
      "Action": "execute-api:Invoke",
      "Resource": "arn:aws:execute-api:us-east-1:111122223333:*/*",
      "Sid": "AllowExecute",
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:CreateModel",
        "sagemaker:CreateEndpointConfig",
        "sagemaker:CreateEndpoint",
        "sagemaker:DescribeModel",
        "sagemaker:DescribeEndpointConfig",
        "sagemaker:DescribeEndpoint",
        "sagemaker:DeleteModel",
        "sagemaker:DeleteEndpointConfig",
        "sagemaker:DeleteEndpoint"
      ],
      "Resource": "arn:aws:sagemaker:us-east-1:111122223333:*/*",
      "Sid": "AllowApiActions",
      "Effect": "Allow"
```

```
    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::111122223333:role/role-name",
      "Sid": "AllowPassRole",
      "Effect": "Allow"
    }
  ]
}
```

10. Enregistrez cette nouvelle politique et attachez-la au rôle IAM à l'étape 8.
11. Sélectionnez Démarrer en haut à droite de la page de détails de l'instance de bloc-notes SageMaker AI pour démarrer l'instance de bloc-notes.

Utilisation du AWS CLI pour configurer Neptune ML sur un cluster de bases de données

Outre le modèle de CloudFormation démarrage rapide et le AWS Management Console, vous pouvez également configurer Neptune ML à l'aide du. AWS CLI

Création d'un groupe de paramètres de cluster de bases de données pour votre nouveau cluster Neptune ML

Les AWS CLI commandes suivantes créent un nouveau groupe de paramètres de cluster de base de données et le configurent pour qu'il fonctionne avec Neptune ML :

Pour créer et configurer un groupe de paramètres de cluster de bases de données pour Neptune ML

1. Créez un nouveau groupe de paramètres de cluster de bases de données :

```
aws neptune create-db-cluster-parameter-group \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --db-parameter-group-family neptune1
  --description "(description of your machine learning project)" \
  --region (AWS region, such as us-east-1)
```

2. Créez un paramètre de `neptune_ml_iam_role` cluster de base de données défini sur l'ARN du `SageMakerExcecutionIAMRole` cluster de base de données à utiliser lorsque vous appelez l' SageMaker IA pour créer des tâches et obtenir des prédictions à partir de modèles de machine learning hébergés :

```
aws neptune modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --parameters "ParameterName=neptune_ml_iam_role, \
                ParameterValue=ARN of the SageMakerExcecutionIAMRole, \
                Description=NeptuneMLRole, \
                ApplyMethod=pending-reboot" \
  --region (AWS region, such as us-east-1)
```

   La définition de ce paramètre permet à Neptune d'accéder à l' SageMaker IA sans que vous ayez à transmettre le rôle à chaque appel.

   Pour en savoir plus sur la façon de créer le `SageMakerExcecutionIAMRole`, consultez Création d'un NeptuneSageMaker IAMRole rôle personnalisé.

3. Enfin, utilisez `describe-db-cluster-parameters` pour vérifier que tous les paramètres du nouveau groupe de paramètres du cluster de bases de données sont définis comme vous le souhaitez :

```
aws neptune describe-db-cluster-parameters \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --region (AWS region, such as us-east-1)
```

Attachement du nouveau groupe de paramètres de cluster de bases de données au cluster de bases de données que vous utiliserez avec Neptune ML

Vous pouvez maintenant attacher le nouveau groupe de paramètres de cluster de bases de données que vous venez de créer à un cluster de bases de données existant à l'aide de la commande suivante :

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (the name of your existing DB cluster) \
  --apply-immediately
  --db-cluster-parameter-group-name (name of your new DB cluster parameter group) \
  --region (AWS region, such as us-east-1)
```

Pour rendre effectifs tous les paramètres, vous pouvez redémarrer le cluster de bases de données :

```
aws neptune reboot-db-instance
  --db-instance-identifier (name of the primary instance of your DB cluster) \
  --profile (name of your AWS profile to use) \
  --region (AWS region, such as us-east-1)
```

Ou, si vous créez un nouveau cluster de bases de données à utiliser avec Neptune ML, vous pouvez utiliser la commande suivante pour créer le cluster avec le nouveau groupe de paramètres attaché, puis créer une nouvelle instance principale (d'enregistreur) :

```
cluster-name=(the name of the new DB cluster)
aws neptune create-db-cluster
  --db-cluster-identifier ${cluster-name}
  --engine graphdb \
  --engine-version 1.0.4.1 \
  --db-cluster-parameter-group-name (name of your new DB cluster parameter group) \
  --db-subnet-group-name (name of the subnet to use) \
```

```
  --region (AWS region, such as us-east-1)

aws neptune create-db-instance
  --db-cluster-identifier ${cluster-name}
  --db-instance-identifier ${cluster-name}-i \
  --db-instance-class (the instance class to use, such as db.r5.xlarge)
  --engine graphdb \
  --region (AWS region, such as us-east-1)
```

Attachez-le **`NeptuneSageMakerIAMRole`** à votre cluster de base de données afin qu'il puisse accéder aux ressources SageMaker AI et Amazon S3

Enfin, suivez les instructions Création d'un NeptuneSageMaker IAMRole rôle personnalisé pour créer un rôle IAM qui permettra à votre cluster de base de données de communiquer avec SageMaker AI et Amazon S3. Ensuite, utilisez la commande suivante pour attacher le rôle `NeptuneSageMakerIAMRole` que vous avez créé à votre cluster de bases de données :

```
aws neptune add-role-to-db-cluster
  --db-cluster-identifier ${cluster-name}
  --role-arn arn:aws:iam::(the ARN number of the role's ARN):role/NeptuneMLRole \
  --region (AWS region, such as us-east-1)
```

Créez deux points de terminaison pour l' SageMaker IA dans votre VPC Neptune

Neptune ML a besoin de deux points de terminaison SageMaker AI dans le VPC de votre cluster de base de données Neptune :

- `com.amazonaws.(AWS region, like us-east-1).sagemaker.runtime`
- `com.amazonaws.(AWS region, like us-east-1).sagemaker.api`

Si vous n'avez pas utilisé le CloudFormation modèle de démarrage rapide, qui les crée automatiquement pour vous, vous pouvez utiliser les AWS CLI commandes suivantes pour les créer :

Celle-ci crée le point de terminaison `sagemaker.runtime` :

```
aws ec2 create-vpc-endpoint
  --vpc-id (the ID of your Neptune DB cluster's VPC)
  --vpc-endpoint-type Interface
  --service-name com.amazonaws.(AWS region, like us-east-1).sagemaker.runtime
  --subnet-ids (the subnet ID or IDs that you want to use)
```

```
  --security-group-ids (the security group for the endpoint network interface, or omit
 to use the default)
  --private-dns-enabled
```

Et celle-ci crée le point de terminaison `sagemaker.api` :

```
aws ec2 create-vpc-endpoint
  --vpc-id (the ID of your Neptune DB cluster's VPC)
  --vpc-endpoint-type Interface
  --service-name com.amazonaws.(AWS region, like us-east-1).sagemaker.api
  --subnet-ids (the subnet ID or IDs that you want to use)
  --security-group-ids (the security group for the endpoint network interface, or omit
 to use the default)
  --private-dns-enabled
```

Vous pouvez également utiliser la console VPC pour créer ces points de terminaison. Consultez Sécuriser les appels de prédiction dans Amazon SageMaker avec AWS PrivateLink et Sécuriser tous les appels SageMaker d'API Amazon avec AWS PrivateLink.

Créez un paramètre de point de terminaison d'inférence SageMaker AI dans le groupe de paramètres de votre cluster de base de données

Pour éviter d'avoir à spécifier le point de terminaison d'inférence SageMaker AI du modèle que vous utilisez dans chaque requête que vous lui envoyez, créez un paramètre de cluster de base de données nommé `neptune_ml_endpoint` dans le groupe de paramètres de cluster de base de données pour Neptune ML. Définissez le paramètre sur l'`id` du point de terminaison de l'instance en question.

Pour ce faire, vous pouvez utiliser la AWS CLI commande suivante :

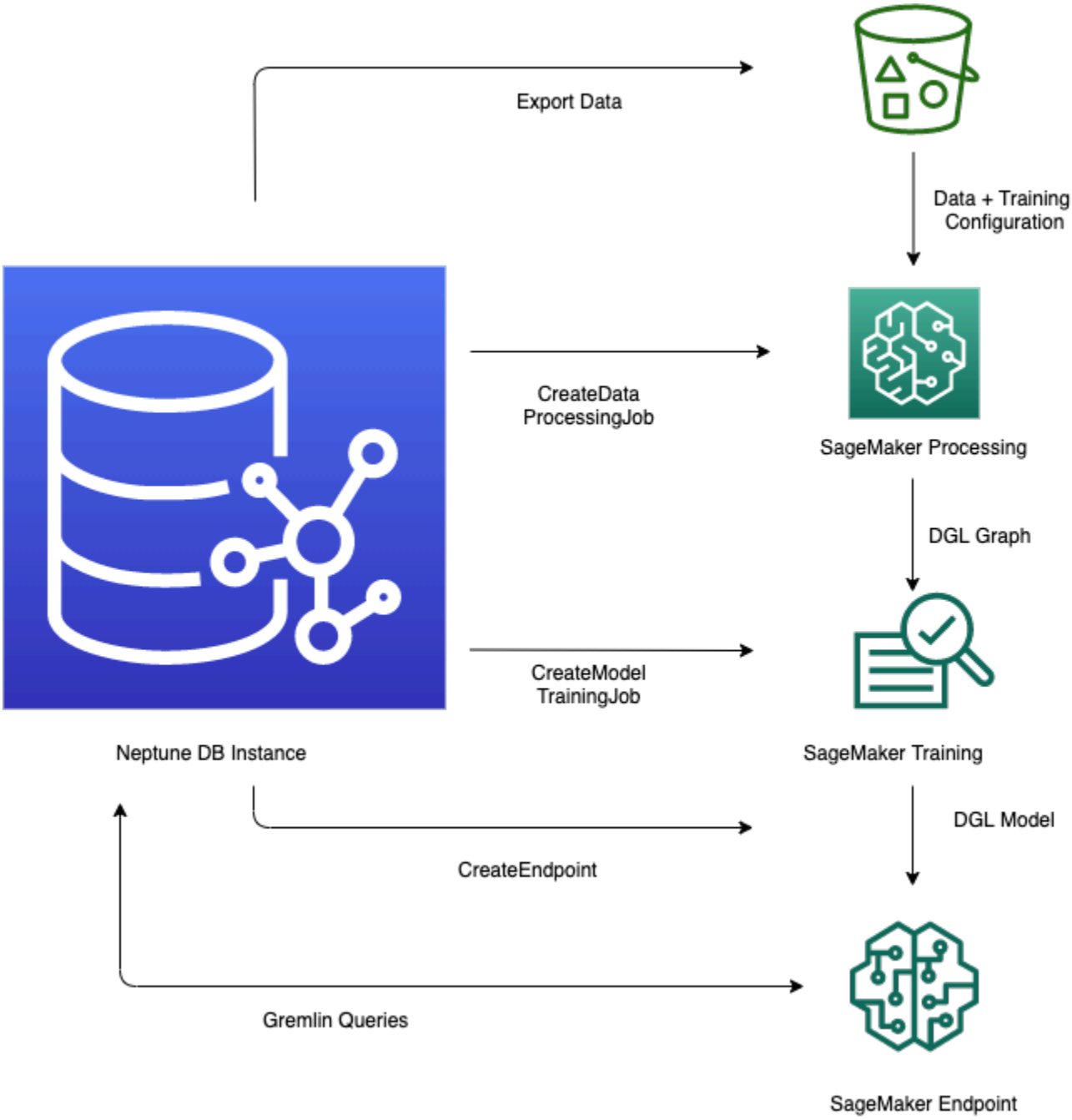
```
aws neptune modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name neptune-ml-demo \
  --parameters "ParameterName=neptune_ml_endpoint, \
                ParameterValue=(the name of the SageMaker AI inference endpoint you
 want to query), \
                Description=NeptuneMLEndpoint, \
                ApplyMethod=pending-reboot" \
  --region (AWS region, such as us-east-1)
```

Vue d'ensemble de la façon d'utiliser la fonctionnalité Neptune ML

La fonctionnalité Neptune ML d'Amazon Neptune fournit un flux de travail rationalisé pour tirer parti des modèles d'apprentissage automatique au sein d'une base de données de graphes. Le processus comprend plusieurs étapes clés : exporter les données de Neptune au format CSV, prétraiter les données pour les préparer à l'entraînement du modèle, entraîner le modèle d'apprentissage automatique à l'aide d'Amazon SageMaker AI, créer un point de terminaison d'inférence pour fournir des prédictions, puis interroger le modèle directement à partir des requêtes Gremlin. Le plan de travail Neptune fournit des commandes magiques pratiques en ligne et en cellule pour faciliter la gestion et l'automatisation de ces étapes. En intégrant des fonctionnalités d'apprentissage automatique directement dans la base de données de graphes, Neptune ML permet aux utilisateurs d'obtenir des informations précieuses et de faire des prédictions à l'aide des riches données relationnelles stockées dans le graphe Neptune.

Flux de travail initial pour l'utilisation de Neptune ML

L'utilisation de la fonctionnalité Neptune ML dans Amazon Neptune implique généralement les cinq étapes suivantes de départ :



1. Exportation et configuration des données : l'étape d'exportation des données utilise le service d'exportation Neptune ou l'outil de ligne de commande `neptune-export` pour exporter les données de Neptune vers Amazon Simple Storage Service (Amazon S3) au format CSV. Un fichier de configuration nommé `training-data-configuration.json` est automatiquement généré au même moment et il indique comment les données exportées peuvent être chargées dans un graphe pouvant être entraîné.

2. Prétraitement des données : dans cette étape, le jeu de données exporté est prétraité à l'aide de techniques standard visant à le préparer pour l'entraînement de modèle. Une normalisation des fonctionnalités peut être effectuée pour les données numériques, et les fonctionnalités de texte peuvent être encodées à l'aide de `word2vec`. À la fin de cette étape, un graphe DGL (bibliothèque Deep Graph) est généré à partir du jeu de données exporté pour l'étape d'entraînement de modèle à utiliser.

   Cette étape est mise en œuvre à l'aide d'une tâche de traitement par SageMaker IA dans votre compte, et les données qui en résultent sont stockées dans un emplacement Amazon S3 que vous avez spécifié.

3. Entraînement de modèle : l'étape d'entraînement de modèle entraîne le modèle de machine learning qui sera utilisé pour les prédictions.

   L'entraînement de modèle se déroule en deux phases :

   - La première étape utilise une tâche de traitement par SageMaker IA pour générer un ensemble de configuration de stratégie d'entraînement du modèle qui spécifie le type de modèle et les plages d'hyperparamètres du modèle qui seront utilisés pour l'apprentissage du modèle.
   - La deuxième étape utilise ensuite une tâche de réglage du modèle SageMaker AI pour tester différentes configurations d'hyperparamètres et sélectionner la tâche de formation qui a produit le modèle le plus performant. La tâche de réglage exécute un nombre prédéfini d'essais de tâches d'entraînement de modèle sur les données traitées. À la fin de cette phase, les paramètres de modèle entraîné de la meilleure tâche d'entraînement sont utilisés pour générer des artefacts de modèle à des fins d'inférence.

4. Création d'un point de terminaison d'inférence dans Amazon SageMaker AI — Le point de terminaison d'inférence est une instance de point de terminaison d' SageMaker IA lancée avec les artefacts du modèle produits par le meilleur travail de formation. Chaque modèle est lié à un seul point de terminaison. Ce point de terminaison est capable d'accepter les demandes entrantes provenant de la base de données orientée graphe et de renvoyer les prédictions de modèle pour les entrées dans les demandes. Une fois que vous avez créé le point de terminaison, celui-ci reste actif jusqu'à ce que vous le supprimiez.

5. Interrogation du modèle de machine learning à l'aide de Gremlin : vous pouvez utiliser des extensions du langage de requête Gremlin pour effectuer des requêtes de prédictions à partir du point de terminaison d'inférence.

> **Note**
>
> Le workbench Neptune contient une magie linéaire et une magie cellulaire qui peuvent vous faire gagner beaucoup de temps lors de la gestion de ces étapes, à savoir :
>
> - %neptune_ml
> - %%%neptune_ml

Réalisation de prédictions basées sur l'évolution des données de graphe

Dans le cas d'un graphe en constante évolution, vous souhaiterez peut-être créer régulièrement de nouvelles prédictions par lots à l'aide de données actualisées. L'interrogation de prédictions précalculées (inférence transductive) peut être nettement plus rapide que la génération de nouvelles prédictions à la volée basées sur les données les plus récentes (inférence inductive). Les deux approches ont leur place, en fonction de la rapidité avec laquelle vos données évoluent et de vos exigences en matière de performances.

Différence entre inférence inductive et inférence transductive

Dans le cadre de l'inférence transductive, Neptune recherche et renvoie des prédictions qui ont été précalculées au moment de l'entraînement.

Dans le cadre de l'inférence inductive, Neptune construit le sous-graphe correspondant et extrait ses propriétés. Le modèle DGL GNN applique ensuite le traitement de données et l'évaluation de modèle en temps réel.

L'inférence inductive peut donc générer des prédictions impliquant des nœuds et des arêtes qui n'étaient pas présents au moment de l'entraînement et qui reflètent l'état actuel du graphe. Cela se fait toutefois au prix d'une latence plus élevée.

Si votre graphe est dynamique, vous pouvez utiliser l'inférence inductive pour être sûr de prendre en compte les données les plus récentes, mais si votre graphe est statique, l'inférence transductive est plus rapide et plus efficace.

L'inférence inductive est désactivée par défaut. Vous pouvez l'activer pour une requête en utilisant le prédicat Gremlin Neptune#ml.inductiveInference dans la requête, comme suit :

```
.with( "Neptune#ml.inductiveInference")
```

Flux de travail transductifs incrémentiels

Lorsque vous mettez à jour les artefacts de modèle en réexécutant simplement les étapes 1 à 3 (de l'exportation et de la configuration des données à la transformation de modèle), Neptune ML propose des méthodes plus simples pour mettre à jour vos prédictions ML par lots à l'aide de nouvelles données. L'une consiste à utiliser un flux de travail basé sur un modèle incrémentiel et une autre consiste à utiliser le réentraînement de modèle avec démarrage à chaud.

Flux de travail basé sur un modèle incrémentiel

Dans ce flux de travail, vous mettez à jour les prédictions ML sans réentraîner le modèle ML.

> **Note**
>
> Vous pouvez le faire uniquement quand les données de graphe ont été mises à jour avec de nouveaux nœuds et/ou arêtes. Cela ne fonctionne pas actuellement quand des nœuds sont supprimés.

1. Exportation et configuration des données : cette étape est identique à celle du flux de travail principal.
2. Prétraitement des données incrémentielles : cette étape est similaire à l'étape de prétraitement des données du flux de travail principal, mais utilise la même configuration de traitement que celle utilisée précédemment, qui correspond à un modèle entraîné spécifique.
3. Transformation de modèle : à la place d'une étape d'entraînement de modèle, cette étape de transformation de modèle prend le modèle entraîné du flux de travail principal et les résultats de l'étape de prétraitement des données incrémentielles, et génère de nouveaux artefacts de modèle à utiliser pour l'inférence. L'étape de transformation du modèle lance une tâche de traitement par SageMaker IA pour effectuer le calcul qui génère les artefacts du modèle mis à jour.
4. Mettre à jour le point de terminaison d'inférence Amazon SageMaker AI — Facultativement, si vous avez un point de terminaison d'inférence existant, cette étape met à jour le point de terminaison avec les nouveaux artefacts du modèle générés par l'étape de transformation du modèle. Vous pouvez également créer un nouveau point de terminaison d'inférence avec les nouveaux artefacts de modèle.

Réentraînement de modèle avec démarrrage à chaud

À l'aide de ce flux de travail, vous pouvez entraîner et déployer un nouveau modèle ML pour effectuer des prédictions à l'aide des données de graphe incrémentielles, mais démarrer à partir d'un modèle existant généré à l'aide du flux de travail principal :

1. Exportation et configuration des données : cette étape est identique à celle du flux de travail principal.
2. Prétraitement des données incrémentielles : cette étape est identique à celle figurant dans le flux de travail d'inférence du modèle incrémentiel. Les nouvelles données de graphe doivent être traitées avec la même méthode de traitement que celle utilisée précédemment pour l'entraînement de modèle.
3. Entraînement de modèle avec démarrage à chaud : l'entraînement de modèle est similaire à ce qui se passe dans le flux de travail principal, mais vous pouvez accélérer la recherche d'hyperparamètres de modèle en exploitant les informations issues de la tâche d'entraînement de modèle précédente.
4. Mettre à jour le point de terminaison d'inférence Amazon SageMaker AI : cette étape est identique à celle du flux de travail d'inférence de modèles incrémentiel.

Flux de travail pour les modèles personnalisés dans Neptune ML

Neptune ML vous permet d'implémenter, d'entraîner et de déployer vos propres modèles personnalisés pour toutes les tâches prises en charge par Neptune ML. Le flux de travail pour le développement et le déploiement d'un modèle personnalisé est essentiellement le même que pour les modèles intégrés, à quelques différences près, comme expliqué dans Flux de travail de modèle personnalisé.

Sélection d'instances pour les phases Neptune ML

Les différentes étapes du traitement de Neptune ML utilisent différentes instances d' SageMaker IA. Ici, nous expliquons comment choisir le type d'instance approprié pour chaque phase. Vous trouverez des informations sur les types d'instances d' SageMaker IA et leurs tarifs sur Amazon SageMaker Pricing.

Sélection d'une instance pour le traitement des données

L'étape de traitement des données par SageMaker IA nécessite une instance de traitement disposant de suffisamment de mémoire et de stockage sur disque pour les données d'entrée, intermédiaires et de sortie. La quantité spécifique de mémoire et de stockage sur disque requise dépend des caractéristiques du graphe Neptune ML et de ses fonctionnalités exportées.

Par défaut, Neptune ML choisit la plus petite instance `ml.r5` dont la mémoire est dix fois supérieure à la taille des données de graphe exportées sur le disque.

Sélection d'une instance pour l'entraînement de modèle et la transformation de modèle

La sélection du type d'instance approprié pour l'entraînement de modèle ou la transformation de modèle dépend du type de tâche, de la taille de graphe et de vos exigences en matière de délai d'exécution. Les instances GPU fournissent les meilleures performances. Nous recommandons généralement les instances en série `p3` et `g4dn`. Vous pouvez également utiliser les instances `p2` ou `p4d`.

Par défaut, Neptune ML choisit la plus petite instance GPU avec plus de mémoire que ce que nécessitent l'entraînement de modèle et la transformation de modèle. Vous pouvez rechercher cette sélection dans le fichier `train_instance_recommendation.json`, dans l'emplacement de sortie du traitement des données Amazon S3. Voici un exemple du contenu d'un fichier `train_instance_recommendation.json` :

```
{
  "instance":     "(the recommended instance type for model training and transform)",
  "cpu_instance": "(the recommended instance type for base processing instance)",
  "disk_size":    "(the estimated disk space required)",
  "mem_size":     "(the estimated memory required)"
}
```

Sélection d'une instance pour un point de terminaison d'inférence

La sélection du type d'instance approprié pour un point de terminaison d'inférence dépend du type de tâche, de la taille de graphe et de votre budget. Par défaut, Neptune ML choisit la plus petite instance `ml.m5d` avec plus de mémoire que celle requise par le point de terminaison d'inférence.

> **Note**
>
> Si plus de 384 Go de mémoire sont nécessaires, Neptune ML utilise une instance `ml.r5d.24xlarge`.

Vous pouvez voir quel type d'instance Neptune ML recommande dans le fichier `infer_instance_recommendation.json` situé dans l'emplacement Amazon S3 que vous utilisez pour l'entraînement de modèle. Voici un exemple de contenu de ce fichier :

```
{
  "instance" :   "(the recommended instance type for an inference endpoint)",
  "disk_size" :  "(the estimated disk space required)",
  "mem_size" :   "(the estimated memory required)"
}
```

Utilisation de l'outil neptune-export ou du service Neptune-Export pour exporter des données depuis Neptune pour Neptune ML

Neptune ML nécessite que vous fournissiez des données d'entraînement pour Deep Graph Library (DGL) afin de créer et d'évaluer des modèles.

Vous pouvez exporter des données depuis Neptune à l'aide du service d'exportation Neptune ou de l'utilitaire `neptune-export`. Le service et l'outil de ligne de commande publient les données dans Amazon Simple Storage Service (Amazon S3) dans un format CSV, chiffrées à l'aide du chiffrement côté serveur Amazon S3 (`SSE-S3`). Consultez Fichiers exportés par Neptune-Export et `neptune-export`.

En outre, lorsque vous configurez une exportation de données d'entraînement pour Neptune ML, la tâche d'exportation crée et publie un fichier de configuration d'entraînement de modèle chiffré avec les données exportées. Par défaut, ce fichier est nommé `training-data-configuration.json`.

Exemples d'utilisation du service d'exportation Neptune pour exporter des données d'entraînement pour Neptune ML

Cette demande exporte les données d'entraînement de graphe de propriétés pour une tâche de classification de nœud :

curl

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-pg",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": {
          "endpoint": "(your Neptune endpoint DNS name)",
          "profile": "neptune_ml"
        },
        "additionalParams": {
          "neptune_ml": {
            "version": "v2.0",
            "targets": [
```

```
            {
              "node": "Movie",
              "property": "genre",
              "type": "classification"
            }
          ]
        }
      }
    }'
```

awscurl

```
awscurl (your NeptuneExportApiUri) \
  --region us-east-1 \
  --service execute-api \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-pg",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": {
          "endpoint": "(your Neptune endpoint DNS name)",
          "profile": "neptune_ml"
        },
        "additionalParams": {
          "neptune_ml": {
            "version": "v2.0",
            "targets": [
              {
                "node": "Movie",
                "property": "genre",
                "type": "classification"
              }
            ]
          }
        }
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Cette demande exporte les données d'entraînement RDF pour une tâche de classification de nœud :

curl

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-rdf",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": {
          "endpoint": "(your Neptune endpoint DNS name)",
          "profile": "neptune_ml"
        },
        "additionalParams": {
          "neptune_ml": {
            "version": "v2.0",
            "targets": [
              {
                "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
                "predicate": "http://aws.amazon.com/neptune/csv2rdf/
datatypeProperty/genre",
                "type": "classification"
              }
            ]
          }
        }
      }'
```

awscurl

```
awscurl (your NeptuneExportApiUri) \
  --region us-east-1 \
  --service execute-api \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-rdf",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": {
          "endpoint": "(your Neptune endpoint DNS name)",
          "profile": "neptune_ml"
```

```
        },
        "additionalParams": {
          "neptune_ml": {
            "version": "v2.0",
            "targets": [
              {
                "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
                "predicate": "http://aws.amazon.com/neptune/csv2rdf/
datatypeProperty/genre",
                "type": "classification"
              }
            ]
          }
        }
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

Champs à définir dans l'objet params lors de l'exportation des données d'entraînement

L'objet d'une demande d'exportation peut contenir différents champs, comme décrit dans la documentation `params`. Les champs suivants sont particulièrement pertinents pour l'exportation de données d'entraînement de machine learning :

- **`endpoint`** : utilisez `endpoint` pour spécifier un point de terminaison d'une instance Neptune dans votre cluster de bases de données que le processus d'exportation peut interroger pour extraire des données.
- **`profile`** : le champ `profile` de l'objet `params` doit être défini sur **`neptune-ml`**.

  Cela permet au processus d'exportation de formater les données exportées de manière appropriée pour l'entraînement de modèle Neptune ML, au format CSV pour des données de graphe de propriétés ou sous forme de N-Triples pour des données RDF. Cela entraîne également la création et l'écriture d'un fichier `training-data-configuration.json` dans le même emplacement Amazon S3 que les données d'entraînement exportées.

- **`cloneCluster`** : s'il est défini sur `true`, le processus d'exportation clone votre cluster de bases de données, exporte depuis le clone, puis supprime le clone une fois terminé.
- **`useIamAuth`** : si l'authentification IAM est activée sur votre cluster de bases de données, vous devez inclure ce champ défini sur `true`.

Le processus d'exportation propose également plusieurs méthodes pour filtrer les données que vous exportez (voir ces exemples).

Utilisation de l'objet AdditionalParams pour régler l'exportation des informations d'entraînement des modèles

L'objet `additionalParams` contient des champs que vous pouvez utiliser pour spécifier des étiquettes de classe de machine learning et des fonctionnalités à des fins d'entraînement et pour guider la création d'un fichier de configuration de données d'entraînement.

Le processus d'exportation ne permet pas de déduire automatiquement quelles propriétés de nœud et d'arête doivent être les étiquettes de classe de machine learning qui doivent servir d'exemples à des fins d'entraînement. Il ne peut pas non plus déduire automatiquement le meilleur encodage des fonctionnalités pour les propriétés numériques, catégorielles et textuelles. Vous devez donc fournir des indications à l'aide des champs de l'objet `additionalParams` pour spécifier ces éléments ou pour remplacer l'encodage par défaut.

Pour des données de graphe de propriétés, la structure de haut niveau d'`additionalParams` dans une demande d'exportation peut ressembler à ceci :

```
{
  "command": "export-pg",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams": {
      "neptune_ml": {
        "version": "v2.0",
        "targets": [ (an array of node and edge class label targets) ],
        "features": [ (an array of node feature hints) ]
    }
  }
```

```
}
```

Pour des données RDF, leur structure de haut niveau peut ressembler à ceci :

```
{
  "command": "export-rdf",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams": {
      "neptune_ml": {
        "version": "v2.0",
        "targets": [ (an array of node and edge class label targets) ]
    }
  }
}
```

Vous pouvez également fournir plusieurs configurations d'exportation à l'aide du champ `jobs` :

```
{
  "command": "export-pg",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams" : {
    "neptune_ml" : {
      "version": "v2.0",
      "jobs": [
        {
          "name" : "(training data configuration name)",
          "targets": [ (an array of node and edge class label targets) ],
          "features": [ (an array of node feature hints) ]
        },
        {
          "name" : "(another training data configuration name)",
          "targets": [ (an array of node and edge class label targets) ],
          "features": [ (an array of node feature hints) ]
        }
      ]
```

```
    }
  }
}
```

Éléments de haut niveau dans le champ neptune_ml dans AdditionalParams

L'élément de version dans neptune_ml

Spécifie la version de configuration des données d'entraînement à générer.

(Facultatif), Type : string, Valeur par défaut : « v2.0 ».

Si vous incluez `version`, réglez-la sur `v2.0`.

Le champ jobs dans neptune_ml

Contient un tableau d'objets de configuration de données d'entraînement, chacun définissant une tâche de traitement de données et contenant :

- **`name`** : nom de la configuration des données d'entraînement à créer.

  Par exemple, une configuration de données d'entraînement portant le nom « job-number-1 » donne lieu à un fichier de configuration de données d'entraînement nommé `job-number-1.json`.
- **`targets`** : tableau JSON de cibles d'étiquettes de classes de nœuds et d'arêtes qui représentent les étiquettes des classes de machine learning à des fins d'entraînement. Consultez Le champ targets dans un objet neptune_ml.
- **`features`** : tableau JSON de fonctionnalités de propriété de nœud. Consultez Le champ des fonctionnalités dans neptune_ml.

Le champ targets dans un objet neptune_ml

Le champ `targets` d'une configuration d'exportation de données d'entraînement JSON contient un tableau d'objets cibles qui spécifient une tâche d'entraînement, ainsi que les étiquettes des classes de machine learning pour l'entraînement de cette tâche. Le contenu des objets cibles varie selon que vous effectuez l'entraînement sur des données de graphe de propriétés ou sur des données RDF.

Pour les tâches de classification et de régression de nœud de graphe de propriétés, les objets cibles du tableau peuvent ressembler à ceci :

```
{
  "node": "(node property-graph label)",
  "property": "(property name)",
  "type" : "(used to specify classification or regression)",
  "split_rate": [0.8,0.2,0.0],
  "separator": ","
}
```

Pour les tâches de prédiction de lien, de régression ou de classification d'arête de graphe de propriétés, elles peuvent ressembler à ceci :

```
{
  "edge": "(edge property-graph label)",
  "property": "(property name)",
  "type" : "(used to specify classification, regression or link_prediction)",
  "split_rate": [0.8,0.2,0.0],
  "separator": ","
}
```

Pour les tâches de régression et de classification RDF, les objets cibles du tableau peuvent ressembler à ceci :

```
{
  "node": "(node type of an RDF node)",
  "predicate": "(predicate IRI)",
  "type" : "(used to specify classification or regression)",
  "split_rate": [0.8,0.2,0.0]
}
```

Pour les tâches de prédiction de lien RDF, les objets cibles du tableau peuvent ressembler à ceci :

```
{
  "subject": "(source node type of an edge)",
  "predicate": "(relation type of an edge)",
  "object": "(destination node type of an edge)",
  "type" : "link_prediction",
  "split_rate": [0.8,0.2,0.0]
}
```

Les objets cibles peuvent contenir les champs suivants :

Table des matières

- Champs figurant dans un objet cible de graphe de propriétés
  - Le champ de nœud (sommet) d'un objet cible
  - Le champ de bord d'un objet cible comportant un graphe de propriétés
  - Le champ de propriété d'un objet cible comportant un graphe de propriétés
  - Le champ de type dans un objet cible comportant un graphe de propriétés
  - Le champ split_rate dans un objet cible Property-Graph
  - Le champ de séparation dans un objet cible comportant un graphe de propriétés
- Champs figurant dans un objet cible RDF
  - Le champ de nœud dans un objet cible RDF
  - Le champ objet d'un objet cible RDF
  - Le champ de prédicat dans un objet cible RDF
  - Le champ d'objet dans un objet cible RDF
  - Le champ de type dans un objet cible RDF
  - Champ split_rate figurant dans un objet cible de graphe de propriétés

Champs figurant dans un objet cible de graphe de propriétés

Le champ de nœud (sommet) d'un objet cible

Étiquette de graphe de propriétés d'un nœud cible (sommet). Un objet cible doit contenir un élément `node` ou un élément `edge`, mais pas les deux.

Un élément `node` peut prendre une valeur unique, comme ceci :

```
"node": "Movie"
```

Ou, dans le cas d'un sommet à plusieurs étiquettes, il peut prendre un tableau de valeurs, comme ceci :

```
"node": ["Content", "Movie"]
```

Le champ de bord d'un objet cible comportant un graphe de propriétés

Spécifie une arête cible à l'aide de sa ou ses étiquettes de nœud de départ, de sa propre étiquette et de sa ou ses étiquettes de nœud de fin. Un objet cible doit contenir un élément `edge` ou un élément `node`, mais pas les deux.

La valeur d'un champ `edge` est un tableau JSON de trois chaînes qui représentent la ou les étiquettes de graphe de propriétés du nœud de départ, l'étiquette de graphe de propriétés de l'arête elle-même, ainsi que la ou les étiquettes de graphe de propriétés du nœud de fin, comme suit :

```
"edge": ["Person_A", "knows", "Person_B"]
```

Si le nœud de départ (nœud de and/or fin) possède plusieurs étiquettes, placez-les dans un tableau, comme ceci :

```
"edge": [ ["Admin", Person_A"], "knows", ["Admin", "Person_B"] ]
```

Le champ de propriété d'un objet cible comportant un graphe de propriétés

Spécifie une propriété du sommet ou de l'arête cible, comme suit :

```
"property" : "rating"
```

Ce champ est obligatoire, sauf lorsque la tâche cible est une prédiction de lien.

Le champ de type dans un objet cible comportant un graphe de propriétés

Indique le type de tâche cible à exécuter sur l'élément `node` ou `edge`, comme ceci :

```
"type" : "regression"
```

Les types de tâche pris en charge pour les nœuds sont les suivants :

- `classification`
- `regression`

Les types de tâche pris en charge pour les arêtes sont les suivants :

- `classification`
- `regression`
- `link_prediction`

Ce champ est obligatoire.

Le champ split_rate dans un objet cible Property-Graph

(Facultatif) Estimation des proportions de nœuds et d'arêtes que les phases d'entraînement, de validation et de test utiliseront, respectivement. Ces proportions sont représentées par un tableau JSON de trois nombres compris entre zéro et un dont la somme est égale à un :

```
"split_rate": [0.7, 0.1, 0.2]
```

Si vous ne fournissez pas le `split_rate` champ facultatif, la valeur estimée par défaut est `[0.9, 0.1, 0.0]` destinée aux tâches de classification et de régression, ainsi qu'`[0.9,0.05, 0.05]`aux tâches de prédiction de liens.

Le champ de séparation dans un objet cible comportant un graphe de propriétés

(Facultatif) Utilisé avec une tâche de classification.

Le champ `separator` spécifie un caractère utilisé pour diviser la valeur d'une propriété cible en plusieurs valeurs catégorielles, lorsqu'il est utilisé pour stocker plusieurs valeurs de catégorie dans une chaîne. Par exemple :

```
"separator": "|"
```

La présence d'un champ `separator` indique que la tâche est une tâche de classification à plusieurs cibles.

Champs figurant dans un objet cible RDF

Le champ de nœud dans un objet cible RDF

Définit le type de nœud des nœuds cibles. Utilisé avec les tâches de classification de nœud ou les tâches de régression de nœud. Le type de nœud d'un nœud dans RDF est défini par :

```
node_id, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, node_type
```

Un élément node RDF peut seulement prendre une valeur unique, comme ceci :

```
"node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

Le champ objet d'un objet cible RDF

Pour les tâches de prédiction de lien, `subject` définit le type de nœud source des arêtes cibles.

```
"subject": "http://aws.amazon.com/neptune/csv2rdf/class/Director"
```

> **Note**
>
> Pour les tâches de prédiction de lien, `subject` doit être utilisé conjointement avec `predicate` et `object`. Si aucune de ces trois options n'est fournie, toutes les arêtes sont traitées comme cible d'entraînement.

Le champ de prédicat dans un objet cible RDF

Pour les tâches de classification de nœud et de régression de nœud, `predicate` définit les données littérales utilisées comme fonctionnalité de nœud cible d'un nœud cible.

```
"predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/genre"
```

> **Note**
>
> Si les nœuds cibles n'ont qu'un seul prédicat définissant la fonctionnalité de nœud cible, le champ `predicate` peut être omis.

Pour les tâches de prédiction de lien, `predicate` définit le type de relation des arêtes cibles :

```
"predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/direct"
```

> **Note**
>
> Pour les tâches de prédiction de lien, `predicate` doit être utilisé conjointement avec `subject` et `object`. Si aucune de ces trois options n'est fournie, toutes les arêtes sont traitées comme cible d'entraînement.

Le champ d'objet dans un objet cible RDF

Pour les tâches de prédiction de lien, `object` définit le type de nœud de destination des arêtes cibles :

```
  "object": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

> **Note**
>
> Pour les tâches de prédiction de lien, `object` doit être utilisé conjointement avec `subject` et `predicate`. Si aucune de ces trois options n'est fournie, toutes les arêtes sont traitées comme cible d'entraînement.

Le champ de type dans un objet cible RDF

Indique le type de tâche cible à effectuer, comme suit :

```
  "type" : "regression"
```

Les types de tâche pris en charge pour les données RDF sont les suivants :

- `link_prediction`
- `classification`
- `regression`

Ce champ est obligatoire.

Champ **`split_rate`** figurant dans un objet cible de graphe de propriétés

(Facultatif) Estimation des proportions de nœuds et d'arêtes que les phases d'entraînement, de validation et de test utiliseront, respectivement. Ces proportions sont représentées par un tableau JSON de trois nombres compris entre zéro et un dont la somme est égale à un :

```
"split_rate": [0.7, 0.1, 0.2]
```

Si vous ne spécifiez pas le champ `split_rate` facultatif, la valeur estimée par défaut est `[0.9, 0.1, 0.0]`.

Le champ des fonctionnalités dans neptune_ml

Les valeurs des propriétés et les littéraux RDF se présentent sous différents formats et types de données. Pour obtenir de bonnes performances de machine learning, il est essentiel de convertir ces valeurs en encodages numériques appelés fonctionnalités.

Neptune ML effectue l'extraction et l'encodage des fonctionnalités dans le cadre des étapes d'exportation et de traitement des données, comme décrit dans Encodage des fonctionnalités dans Neptune ML.

Pour les jeux de données de graphe de propriétés, le processus d'exportation déduit automatiquement les fonctionnalités `auto` pour les propriétés de chaîne et pour les propriétés numériques qui contiennent des valeurs multiples. Pour les propriétés numériques contenant des valeurs uniques, il déduit des fonctionnalités `numerical`. Pour les propriétés de date, il déduit des fonctionnalités `datetime`.

Si vous souhaitez remplacer une spécification de fonction déduite automatiquement ou ajouter une spécification numérique de compartiment, TF-IDF ou SBERT pour une propriété FastText, vous pouvez contrôler le codage des entités à l'aide du champ Features.

> **Note**
>
> Vous pouvez uniquement utiliser le champ `features` pour contrôler les spécifications des fonctionnalités pour les données de graphe de propriétés, et non pour les données RDF.

Pour le texte de forme libre, Neptune ML peut utiliser plusieurs modèles différents pour convertir la séquence de jetons d'une valeur de propriété de chaîne en un vecteur de valeur réelle de taille fixe :

- `text_fasttext` : utilise l'encodage fastText. Il s'agit de l'encodage recommandé pour les fonctionnalités qui utilisent une et une seule des cinq langues prises en charge par fastText.
- `text_sbert` : utilise les modèles d'encodage Sentence BERT (SBERT). Il s'agit de l'encodage recommandé pour le texte que `text_fasttext` ne prend pas en charge.
- `text_word2vec` : utilise les algorithmes Word2Vec publiés à l'origine par Google pour encoder du texte. Word2Vec prend en charge uniquement l'anglais.
- `text_tfidf` : utilise un vectoriseur de fréquence de terme et de fréquence de document inverse (TF-IDF) pour l'encodage de texte. L'encodage TF-IDF prend en charge des fonctionnalités statistiques que les autres encodages ne prennent pas en charge.

Le champ `features` contient un tableau JSON des fonctionnalités des propriétés des nœuds. Les objets figurant dans le tableau peuvent contenir les champs suivants :

Table des matières

- Le champ de nœuds dans les fonctionnalités
- Le champ périphérique dans les fonctionnalités
- Le champ de propriété dans les fonctionnalités
- Valeurs possibles du champ de type pour les entités
- Le champ des normes
- Le champ de langue
- Le champ max_length
- Le champ du séparateur
- Le champ de portée
- Le champ bucket_cnt
- Le champ slide_window_size
- Le champ de l'imputeur
- Le champ max_features
- Le champ min_df
- Le champ ngram_range
- Le champ datetime_parts

Le champ de nœuds dans les fonctionnalités

Le champ `node` spécifie l'étiquette de graphe de propriétés d'un sommet de fonctionnalité. Par exemple :

```
  "node": "Person"
```

Si un sommet possède plusieurs étiquettes, utilisez un tableau qui les contiendra. Par exemple :

```
  "node": ["Admin", "Person"]
```

Le champ périphérique dans les fonctionnalités

Le champ `edge` spécifie le type d'arête d'une arête de fonctionnalité. Un type d'arête consiste en un tableau contenant les étiquettes de graphe de propriétés du sommet source, l'étiquette de graphe de propriétés de l'arête et la ou les étiquettes de graphe de propriétés du sommet de destination. Vous devez fournir ces trois valeurs lorsque vous spécifiez une fonctionnalité d'arête. Par exemple :

```
"edge": ["User", "reviewed", "Movie"]
```

Si un sommet source ou de destination d'un type d'arête possède plusieurs étiquettes, utilisez un autre tableau qui les contiendra. Par exemple :

```
"edge": [["Admin", "Person"]. "edited", "Post"]
```

Le champ de propriété dans les fonctionnalités

Utilisez le paramètre de propriété pour spécifier une propriété du sommet identifié par le paramètre `node`. Par exemple :

```
"property" : "age"
```

Valeurs possibles du champ de type pour les entités

Le paramètre `type` indique le type de fonctionnalité en cours de définition. Par exemple :

```
"type": "bucket_numerical"
```

Valeurs possibles du paramètre **`type`**

- **`"auto"`** : spécifie que Neptune ML doit détecter automatiquement le type de propriété et appliquer un encodage de fonctionnalité approprié. Une fonctionnalité `auto` peut également comporter un champ `separator` facultatif.

  Consultez Encodage automatique des fonctionnalités dans Neptune ML.
- **`"category"`** : cet encodage de fonctionnalité représente une valeur de propriété sous la forme d'une catégorie parmi un certain nombre de catégories. En d'autres termes, la fonctionnalité peut prendre une ou plusieurs valeurs discrètes. Une fonctionnalité `category` peut également comporter un champ `separator` facultatif.

Consultez Fonctionnalités catégorielles dans Neptune ML.

- **`"numerical"`** : cet encodage de fonctionnalité représente les valeurs de propriété numériques sous forme de nombres dans un intervalle continu où « supérieur à » et « inférieur à » ont une signification.

  Une fonctionnalité `numerical` peut également comporter des champs `norm`, `imputer` et `separator` facultatifs.

  Consultez Fonctionnalités numériques dans Neptune ML.

- **`"bucket_numerical"`** : cet encodage de fonctionnalité divise les valeurs de propriété numériques en un ensemble de compartiments ou de catégories.

  Par exemple, vous pouvez encoder l'âge des personnes dans 4 compartiments : les enfants (0 à 20), les jeunes adultes (20 à 40), les personnes d'âge moyen (40 à 60) et les personnes âgées (à partir de 60).

  Une `bucket_numerical` fonctionnalité nécessite un `range` `bucket_cnt` champ et peut éventuellement inclure un `imputer` and/or `slide_window_size` champ.

  Consultez Fonctionnalités numériques de compartiment dans Neptune ML.

- **`"datetime"`** : cet encodage de fonctionnalité représente une valeur de propriété datetime sous la forme d'un tableau de ces fonctionnalités catégorielles : année, mois, jour de la semaine et heure.

  Une ou plusieurs de ces quatre catégories peuvent être éliminées à l'aide du paramètre `datetime_parts`.

  Consultez Fonctionnalités datetime dans Neptune ML.

- **`"text_fasttext"`** : cet encodage de fonctionnalité convertit les valeurs de propriété constituées de phrases ou de texte libre en vecteurs numériques à l'aide de modèles fastText. Il prend en charge cinq langues, à savoir l'anglais (`en`), le chinois (`zh`), l'hindi (`hi`), l'espagnol (`es`) et le français (`fr`). Pour les valeurs de propriété de texte dans l'une de ces cinq langues, `text_fasttext` est l'encodage recommandé. Toutefois, il ne peut pas traiter les cas où une même phrase contient des mots dans plusieurs langues.

  Pour les langues autres que celles prises en charge par fastText, utilisez l'encodage `text_sbert`.

Si vous avez de nombreuses chaînes de texte de valeur de propriété de plus de 120 jetons, par exemple, utilisez le champ `max_length` pour limiter le nombre de jetons dans chaque chaîne encodée par `"text_fasttext"`.

Consultez Encodage fastText des valeurs de propriété de texte dans Neptune ML.

- **`"text_sbert"`** : cet encodage convertit les valeurs de propriété de texte en vecteurs numériques à l'aide des modèles Sentence BERT (SBERT). Neptune prend en charge deux méthodes SBERT, à savoir `text_sbert128`, qui est la méthode par défaut si vous spécifiez simplement `text_sbert`, et `text_sbert512`. La différence entre les deux est le nombre maximal de jetons qui sont encodés dans une propriété de texte. L'encodage `text_sbert128` encode uniquement les 128 premiers jetons, tandis que `text_sbert512` encode jusqu'à 512 jetons. Par conséquent, l'utilisation de `text_sbert512` peut nécessiter plus de temps de traitement que `text_sbert128`. Les deux méthodes sont plus lentes que `text_fasttext`.

  Les méthodes `text_sbert*` prennent en charge de nombreuses langues et peuvent encoder une phrase contenant plusieurs langues.

  Consultez Encodage Sentence BERT (SBERT) des fonctionnalités de texte dans Neptune ML.

- **`"text_word2vec"`** : cet encodage convertit les valeurs de propriété de texte en vecteurs numériques à l'aide des algorithmes Word2Vec. Il prend en charge uniquement l'anglais.

  Consultez Encodage Word2Vec des fonctionnalités textuelles dans Neptune ML.

- **`"text_tfidf"`** : cet encodage convertit les valeurs de propriété de texte en vecteurs numériques à l'aide d'un vectoriseur de fréquence de terme et de fréquence de document inverse (TF-IDF).

  Vous définissez les paramètres d'un encodage de fonctionnalité `text_tfidf` à l'aide du champ `ngram_range`, du champ `min_df` et du champ `max_features`.

  Consultez Encodage TF-IDF des fonctionnalités de texte dans Neptune ML.

- **`"none"`** : l'utilisation du type `none` n'entraîne aucun encodage des fonctionnalités. Les valeurs de propriété brutes sont analysées et enregistrées à la place.

  Utilisez `none` uniquement si vous envisagez d'effectuer votre propre encodage de fonctionnalités personnalisé dans le cadre de l'entraînement de modèle personnalisé.

Le champ des normes

Ce champ est obligatoire pour les fonctionnalités numériques. Il spécifie une méthode de normalisation à utiliser sur les valeurs numériques :

```
"norm": "min-max"
```

Les méthodes de normalisation suivantes sont prises en charge :

- « min-max » : normalisez chaque valeur en lui soustrayant la valeur minimale, puis en la divisant par la différence entre la valeur maximale et la valeur minimale.
- « standard » : normalisez chaque valeur en la divisant par la somme de toutes les valeurs.
- « aucune » : ne normalisez pas les valeurs numériques pendant l'encodage.

Consultez Fonctionnalités numériques dans Neptune ML.

Le champ de langue

Le champ de langue indique la langue utilisée dans les valeurs des propriétés de texte. Son utilisation dépend de la méthode d'encodage de texte :

- Pour l'encodage `text_fasttext`, ce champ est obligatoire et doit spécifier l'une des langues suivantes :
  - `en` (anglais)
  - `zh` (chinois)
  - `hi` (hindi)
  - `es` (espagnol)
  - `fr` (français)
- Pour l'encodage `text_sbert`, ce champ n'est pas utilisé, car l'encodage SBERT est multilingue.
- Pour l'encodage `text_word2vec`, ce champ est facultatif, car `text_word2vec` prend en charge uniquement l'anglais. Le cas échéant, il doit spécifier le nom du modèle de langue anglaise :

  ```
  "language" : "en_core_web_lg"
  ```

- Pour l'encodage `text_tfidf`, ce champ n'est pas utilisé.

Le champ max_length

Le champ `max_length` est facultatif pour les fonctionnalités `text_fasttext`, où il indique le nombre maximal de jetons qui seront encodés dans une fonctionnalité de texte en entrée. Entrez du texte d'une longueur supérieure à celle de `max_length` qui est tronquée. Par exemple, définir max_length sur 128 indique que tous les jetons situés après le 128e dans une séquence de texte seront ignorés :

```
"max_length": 128
```

Le champ du séparateur

Ce champ est utilisé en option avec les fonctionnalités `category`, `numerical` et `auto`. Il spécifie un caractère qui peut être utilisé pour diviser une valeur de propriété en plusieurs valeurs catégorielles ou valeurs numériques :

```
"separator": ";"
```

Utilisez le champ `separator` seulement quand la propriété stocke plusieurs valeurs délimitées dans une seule chaîne, telle que `"Actor;Director"` ou `"0.1;0.2"`.

Consultez Fonctionnalités catégorielles, Fonctionnalités numériques et Encodage automatique.

Le champ de portée

Ce champ est obligatoire pour les fonctionnalités `bucket_numerical`. Il indique la plage de valeurs numériques qui doivent être divisées en compartiments, au format `[`*`lower-bound`*`, `*`upper-bound`*`]` :

```
"range" : [20, 100]
```

Si une valeur de propriété est inférieure à la limite inférieure, elle est affectée au premier compartiment, ou si elle est supérieure à la limite supérieure, elle est affectée au dernier compartiment.

Consultez Fonctionnalités numériques de compartiment dans Neptune ML.

Le champ bucket_cnt

Ce champ est obligatoire pour les fonctionnalités `bucket_numerical`. Il spécifie le nombre de compartiments dans lesquels la plage numérique définie par le paramètre `range` doit être divisée :

```
"bucket_cnt": 10
```

Consultez Fonctionnalités numériques de compartiment dans Neptune ML.

Le champ slide_window_size

Ce champ est utilisé en option avec les fonctionnalités `bucket_numerical` pour attribuer des valeurs à plusieurs compartiments :

```
"slide_window_size": 5
```

Le fonctionnement d'une fenêtre coulissante est le suivant : Neptune ML prend la taille de la fenêtre **s** et transforme chaque valeur numérique **v** d'une propriété en une plage allant de `v - s/2` à `v + s/2` . La valeur est ensuite attribuée à chaque compartiment que la plage chevauche.

Consultez Fonctionnalités numériques de compartiment dans Neptune ML.

Le champ de l'imputeur

Ce champ est utilisé en option avec les fonctionnalités `numerical` et `bucket_numerical` pour fournir une technique d'imputation permettant de renseigner les valeurs manquantes :

```
"imputer": "mean"
```

Les techniques d'imputation prises en charge sont les suivantes :

- `"mean"`
- `"median"`
- `"most-frequent"`

Si vous n'incluez pas le paramètre imputer, le prétraitement des données s'arrête lorsqu'une valeur manquante est détectée.

Consultez Fonctionnalités numériques dans Neptune ML et Fonctionnalités numériques de compartiment dans Neptune ML.

Le champ max_features

Ce champ est utilisé en option par les fonctionnalités `text_tfidf` pour spécifier le nombre maximal de termes à encoder :

```
"max_features": 100
```

Avec un paramètre de 100, le vectoriseur TF-IDF encode uniquement les 100 termes les plus courants. Si vous n'incluez pas `max_features`, la valeur par défaut est 5 000.

Consultez Encodage TF-IDF des fonctionnalités de texte dans Neptune ML.

Le champ min_df

Ce champ est utilisé en option par les fonctionnalités `text_tfidf` pour spécifier la fréquence minimale des documents des termes à encoder :

```
"min_df": 5
```

Un paramètre de 5 indique qu'un terme doit apparaître dans au moins 5 valeurs de propriété différentes pour être encodé.

La valeur par défaut si vous n'incluez pas le paramètre `min_df` est 2.

Consultez Encodage TF-IDF des fonctionnalités de texte dans Neptune ML.

Le champ ngram_range

Ce champ est utilisé en option par les fonctionnalités `text_tfidf` pour spécifier la taille des séquences de mots ou de jetons à considérer comme des termes individuels potentiels à encoder :

```
"ngram_range": [2, 4]
```

La valeur `[2, 4]` indique que les séquences de 2, 3 et 4 mots doivent être considérées comme des termes individuels potentiels.

Si vous ne définissez pas explicitement `ngram_range`, la valeur par défaut est `[1, 1]`, ce qui signifie que seuls des mots ou des jetons uniques sont considérés comme des termes à encoder.

Consultez Encodage TF-IDF des fonctionnalités de texte dans Neptune ML.

Le champ datetime_parts

Ce champ est utilisé en option par les fonctionnalités `datetime` pour spécifier les parties de la valeur datetime à encoder de manière catégorielle :

```
"datetime_parts": ["weekday", "hour"]
```

Si vous n'incluez pas `datetime_parts`, Neptune ML encode par défaut les parties année, mois, jour de la semaine et heure de la valeur datetime. La valeur `["weekday", "hour"]` indique que seuls le jour de la semaine et l'heure des valeurs datetime doivent être encodés de manière catégorielle dans la fonctionnalité.

Si l'une des parties ne possède pas plus d'une valeur unique dans le jeu d'entraînement, elle n'est pas encodée.

Consultez Fonctionnalités datetime dans Neptune ML.

Exemples d'utilisation de paramètres dans AdditionalParams pour ajuster la configuration d'entraînement des modèles

Les exemples suivants montrent comment utiliser la fonctionnalité « AdditionalParams » dans les modèles de données Property-Graph et RDF pour configurer différents aspects du processus d'apprentissage des modèles pour une application Neptune ML. Les exemples couvrent un large éventail de fonctionnalités, notamment la spécification des taux de division par défaut pour les training/validation/test données, la définition de tâches de classification des nœuds, de régression et de prédiction de liens, ainsi que la configuration de différents types de fonctionnalités tels que les compartiments numériques, les intégrations de texte, la date et l'heure et les données catégorielles. Ces configurations détaillées vous permettent d'adapter le pipeline d'apprentissage automatique à vos exigences spécifiques en matière de données et de modélisation, libérant ainsi tout le potentiel des fonctionnalités de Neptune ML.

Table des matières

- Exemples de graphes de propriétés utilisant AdditionalParams
  - Spécification d'un taux de division par défaut pour la configuration d'entraînement de modèle
  - Spécification d'une tâche de classification de nœud pour la configuration d'entraînement de modèle
  - Spécification d'une tâche de classification de nœud multiclasse pour la configuration d'entraînement de modèle
  - Spécification d'une tâche de régression de nœud pour la configuration d'entraînement de modèle
  - Spécification d'une tâche de classification d'arête pour la configuration d'entraînement de modèle
  - Spécification d'une tâche de classification d'arête multiclasse pour la configuration d'entraînement de modèle
  - Spécification d'une régression d'arête pour la configuration d'entraînement de modèle
  - Spécification d'une tâche de prédiction de lien pour la configuration d'entraînement de modèle
  - Spécification d'une fonctionnalité de compartiment numérique
  - Spécification d'une fonctionnalité Word2Vec
  - Spécification d'une fonctionnalité FastText
  - Spécification d'une fonctionnalité Sentence BERT
  - Spécification d'une fonctionnalité TF-IDF
  - Spécification d'une fonctionnalité datetime

- Spécification d'une fonctionnalité category
- Spécification d'une fonctionnalité numerical
- Spécification d'une fonctionnalité auto
- Exemples RDF utilisant additionalParams
  - Spécification d'un taux de division par défaut pour la configuration d'entraînement de modèle
  - Spécification d'une tâche de classification de nœud pour la configuration d'entraînement de modèle
  - Spécification d'une tâche de régression de nœud pour la configuration d'entraînement de modèle
  - Spécification d'une tâche de prédiction de lien pour des arêtes spécifiques
  - Spécification d'une tâche de prédiction de lien pour toutes les arêtes

Exemples de graphes de propriétés utilisant AdditionalParams

Spécification d'un taux de division par défaut pour la configuration d'entraînement de modèle

Dans l'exemple suivant, le paramètre `split_rate` définit le taux de division par défaut pour l'entraînement de modèle. Si aucun taux de division par défaut n'est spécifié, l'entraînement utilise une valeur de [0.9, 0.1, 0.0]. Vous pouvez remplacer la valeur par défaut pour chaque cible en spécifiant un `split_rate` pour chaque cible.

Dans l'exemple suivant, le champ `default split_rate` indique qu'un taux de division de `[0.7,0.1,0.2]` doit être utilisé, sauf s'il est remplacé cible par cible :

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
    "targets": [
      (...)
    ],
    "features": [
      (...)
    ]
  }
}
```

Spécification d'une tâche de classification de nœud pour la configuration d'entraînement de modèle

Pour indiquer quelle propriété de nœud contient des exemples étiquetés à des fins d'entraînement, ajoutez un élément de classification de nœud au tableau `targets` en utilisant `"type" : "classification"`. Ajoutez un champ `split_rate` si vous souhaitez remplacer le taux de division par défaut.

Dans l'exemple suivant, la cible `node` indique que la propriété `genre` de chaque nœud `Movie` doit être traitée comme une étiquette de classe de nœuds. La valeur `split_rate` remplace le taux de division par défaut :

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "genre",
        "type": "classification",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Spécification d'une tâche de classification de nœud multiclasse pour la configuration d'entraînement de modèle

Pour indiquer quelle propriété de nœud contient plusieurs exemples étiquetés à des fins d'entraînement, ajoutez un élément de classification de nœud au tableau de cibles, en utilisant `"type" : "classification"`, et `separator` pour spécifier un caractère qui peut être utilisé pour diviser la valeur d'une propriété cible en plusieurs valeurs catégorielles. Ajoutez un champ `split_rate` si vous souhaitez remplacer le taux de division par défaut.

Dans l'exemple suivant, la cible `node` indique que la propriété `genre` de chaque nœud `Movie` doit être traitée comme une étiquette de classe de nœuds. Le champ `separator` indique que chaque propriété de genre contient plusieurs valeurs séparées par des points-virgules :

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "genre",
        "type": "classification",
        "separator": ";"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Spécification d'une tâche de régression de nœud pour la configuration d'entraînement de modèle

Pour indiquer quelle propriété de nœud contient des régressions étiquetées à des fins d'entraînement, ajoutez un élément de régression de nœud au tableau de cibles, en utilisant `"type" : "regression"`. Ajoutez un champ split_rate si vous souhaitez remplacer le taux de division par défaut.

La cible node suivante indique que la propriété `rating` de chaque nœud `Movie` doit être traitée comme une étiquette de régression de nœud :

```
    "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "rating",
        "type" : "regression",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
    "features": [
      ...
    ]
  }
```

```
}
```

Spécification d'une tâche de classification d'arête pour la configuration d'entraînement de modèle

Pour indiquer quelle propriété d'arête contient des exemples étiquetés à des fins d'entraînement, ajoutez un élément d'arête au tableau `targets` en utilisant `"type" : "regression"`. Ajoutez un champ split_rate si vous souhaitez remplacer le taux de division par défaut.

La cible `edge` suivante indique que la propriété `metAtLocation` de chaque arête `knows` doit être traitée comme une étiquette de classe d'arêtes.

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "knows", "Person"],
        "property": "metAtLocation",
        "type": "classification"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Spécification d'une tâche de classification d'arête multiclasse pour la configuration d'entraînement de modèle

Pour indiquer quelle propriété d'arête contient plusieurs exemples étiquetés à des fins d'entraînement, ajoutez un élément d'arête au tableau `targets`, en utilisant `"type" : "classification"`, et un champ `separator` pour spécifier un caractère utilisé pour diviser une valeur de propriété cible en plusieurs valeurs catégorielles. Ajoutez un champ `split_rate` si vous souhaitez remplacer le taux de division par défaut.

La cible `edge` suivante indique que la propriété `sentiment` de chaque arête `repliedTo` doit être traitée comme une étiquette de classe d'arêtes : Le champ de séparation indique que chaque propriété de sentiment contient plusieurs valeurs séparées par des virgules :

```
  "additionalParams": {
```

```
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "repliedTo", "Message"],
        "property": "sentiment",
        "type": "classification",
        "separator": ","
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Spécification d'une régression d'arête pour la configuration d'entraînement de modèle

Pour indiquer quelle propriété d'arête contient des exemples de régression étiquetés à des fins d'entraînement, ajoutez un élément `edge` au tableau `targets` en utilisant `"type" : "regression"`. Ajoutez un champ `split_rate` si vous souhaitez remplacer le taux de division par défaut.

La cible `edge` suivante indique que la propriété `rating` de chaque arête `reviewed` doit être traitée comme une régression d'arête :

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "reviewed", "Movie"],
        "property": "rating",
        "type" : "regression"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Spécification d'une tâche de prédiction de lien pour la configuration d'entraînement de modèle

Pour indiquer quelles arêtes doivent être utilisées à des fins d'entraînement de prédiction de lien, ajoutez un élément d'arête au tableau de cibles en utilisant `"type" : "link_prediction"`. Ajoutez un champ `split_rate` si vous souhaitez remplacer le taux de division par défaut.

La cible `edge` suivante indique que les arêtes `cites` doivent être utilisées pour la prédiction de lien :

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Article", "cites", "Article"],
        "type" : "link_prediction"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Spécification d'une fonctionnalité de compartiment numérique

Vous pouvez spécifier une fonctionnalité de données numériques pour une propriété de nœud en ajoutant `"type": "bucket_numerical"` au tableau `features`.

La fonctionnalité `node` suivante indique que la propriété `age` de chaque nœud `Person` doit être traitée comme une fonctionnalité de compartiment numérique :

```
  "additionalParams": {
  "neptune_ml": {
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Person",
        "property": "age",
        "type": "bucket_numerical",
        "range": [1, 100],
        "bucket_cnt": 5,
```

```
        "slide_window_size": 3,
        "imputer": "median"
      }
    ]
  }
}
```

Spécification d'une fonctionnalité **`Word2Vec`**

Vous pouvez spécifier une fonctionnalité `Word2Vec` pour une propriété de nœud en ajoutant `"type": "text_word2vec"` au tableau `features`.

La fonctionnalité node suivante indique que la propriété `description` de chaque nœud `Movie` doit être traitée comme une fonctionnalité `Word2Vec` :

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_word2vec",
        "language": "en_core_web_lg"
      }
    ]
  }
}
```

Spécification d'une fonctionnalité **`FastText`**

Vous pouvez spécifier une fonctionnalité `FastText` pour une propriété de nœud en ajoutant `"type": "text_fasttext"` au tableau `features`. Le champ `language` est obligatoire et doit spécifier l'un des codes de langue suivants :

- `en` (anglais)
- `zh` (chinois)
- `hi` (hindi)

- `es` (espagnol)
- `fr` (français)

Notez que l'encodage `text_fasttext` ne peut pas gérer plus d'une langue à la fois dans une fonctionnalité.

La fonctionnalité node suivante indique que la propriété `description` française de chaque nœud `Movie` doit être traitée comme une fonctionnalité `FastText` :

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_fasttext",
        "language": "fr",
        "max_length": 1024
      }
    ]
  }
}
```

Spécification d'une fonctionnalité **`Sentence BERT`**

Vous pouvez spécifier une fonctionnalité `Sentence BERT` pour une propriété de nœud en ajoutant `"type": "text_sbert"` au tableau `features`. Il n'est pas nécessaire de spécifier la langue, car la méthode encode automatiquement les fonctionnalités de texte à l'aide d'un modèle de langue multilingue.

La fonctionnalité node suivante indique que la propriété `description` de chaque nœud `Movie` doit être traitée comme une fonctionnalité `Sentence BERT` :

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
```

```
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_sbert128",
      }
    ]
  }
}
```

Spécification d'une fonctionnalité **`TF-IDF`**

Vous pouvez spécifier une fonctionnalité `TF-IDF` pour une propriété de nœud en ajoutant `"type": "text_tfidf"` au tableau `features`.

La fonctionnalité node suivante indique que la propriété `bio` de chaque nœud `Person` doit être traitée comme une fonctionnalité `TF-IDF` :

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "bio",
        "type": "text_tfidf",
        "ngram_range": [1, 2],
        "min_df": 5,
        "max_features": 1000
      }
    ]
  }
}
```

Spécification d'une fonctionnalité **`datetime`**

Le processus d'exportation déduit automatiquement des fonctionnalités `datetime` pour les propriétés de date. Toutefois, si vous souhaitez limiter l'élément `datetime_parts` utilisé pour une

fonctionnalité `datetime` ou remplacer une spécification de fonctionnalité afin qu'une propriété qui serait normalement traitée comme une fonctionnalité `auto` soit explicitement traitée comme une fonctionnalité `datetime`, vous pouvez le faire en ajoutant un élément `"type": "datetime"` au tableau de fonctionnalités.

La fonctionnalité `node` suivante indique que la propriété `createdAt` de chaque nœud `Post` doit être traitée comme une fonctionnalité `datetime` :

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Post",
        "property": "createdAt",
        "type": "datetime",
        "datetime_parts": ["month", "weekday", "hour"]
      }
    ]
  }
}
```

Spécification d'une fonctionnalité **`category`**

Le processus d'exportation déduit automatiquement des fonctionnalités `auto` pour les propriétés de chaîne et pour les propriétés numériques qui contiennent des valeurs multiples. Pour les propriétés numériques contenant des valeurs uniques, il déduit des fonctionnalités `numerical`. Pour les propriétés de date, il déduit des fonctionnalités `datetime`.

Si vous souhaitez remplacer une spécification de fonctionnalité afin qu'une propriété soit traitée comme une fonctionnalité catégorielle, ajoutez un élément `"type": "category"` au tableau de fonctionnalités. Si la propriété contient plusieurs valeurs, incluez un champ `separator`. Par exemple :

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
```

```
      ...
    ],
    "features": [
      {
        "node": "Post",
        "property": "tag",
        "type": "category",
        "separator": "|"
      }
    ]
  }
}
```

Spécification d'une fonctionnalité **`numerical`**

Le processus d'exportation déduit automatiquement des fonctionnalités `auto` pour les propriétés de chaîne et pour les propriétés numériques qui contiennent des valeurs multiples. Pour les propriétés numériques contenant des valeurs uniques, il déduit des fonctionnalités `numerical`. Pour les propriétés de date, il déduit des fonctionnalités `datetime`.

Si vous souhaitez remplacer une spécification de fonctionnalité afin qu'une propriété soit traitée comme une fonctionnalité `numerical`, ajoutez un élément `"type": "numerical"` au tableau de fonctionnalités. Si la propriété contient plusieurs valeurs, incluez un champ `separator`. Par exemple :

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Recording",
        "property": "duration",
        "type": "numerical",
        "separator": ","
      }
    ]
  }
}
```

Spécification d'une fonctionnalité **`auto`**

Le processus d'exportation déduit automatiquement des fonctionnalités `auto` pour les propriétés de chaîne et pour les propriétés numériques qui contiennent des valeurs multiples. Pour les propriétés numériques contenant des valeurs uniques, il déduit des fonctionnalités `numerical`. Pour les propriétés de date, il déduit des fonctionnalités `datetime`.

Si vous souhaitez remplacer une spécification de fonctionnalité afin qu'une propriété soit traitée comme une fonctionnalité `auto`, ajoutez un élément `"type": "auto"` au tableau de fonctionnalités. Si la propriété contient plusieurs valeurs, incluez un champ `separator`. Par exemple :

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "User",
        "property": "role",
        "type": "auto",
        "separator": ","
      }
    ]
  }
}
```

Exemples RDF utilisant **`additionalParams`**

Spécification d'un taux de division par défaut pour la configuration d'entraînement de modèle

Dans l'exemple suivant, le paramètre `split_rate` définit le taux de division par défaut pour l'entraînement de modèle. Si aucun taux de division par défaut n'est spécifié, l'entraînement utilise une valeur de [0.9, 0.1, 0.0]. Vous pouvez remplacer la valeur par défaut pour chaque cible en spécifiant un `split_rate` pour chaque cible.

Dans l'exemple suivant, le champ `default split_rate` indique qu'un taux de division de `[0.7,0.1,0.2]` doit être utilisé, sauf s'il est remplacé cible par cible :

```
  "additionalParams": {
```

```
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
    "targets": [
      (...)
    ]
  }
}
```

Spécification d'une tâche de classification de nœud pour la configuration d'entraînement de modèle

Pour indiquer quelle propriété de nœud contient des exemples étiquetés à des fins d'entraînement, ajoutez un élément de classification de nœud au tableau `targets` en utilisant `"type" : "classification"`. Ajoutez un champ de nœud pour indiquer le type de nœud des nœuds cibles. Ajoutez un champ `predicate` pour définir les données littérales utilisées comme fonctionnalité de nœud cible du nœud cible. Ajoutez un champ `split_rate` si vous souhaitez remplacer le taux de division par défaut.

Dans l'exemple suivant, la cible `node` indique que la propriété `genre` de chaque nœud `Movie` doit être traitée comme une étiquette de classe de nœuds. La valeur `split_rate` remplace le taux de division par défaut :

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/genre",
        "type": "classification",
        "split_rate": [0.7,0.1,0.2]
      }
    ]
  }
}
```

Spécification d'une tâche de régression de nœud pour la configuration d'entraînement de modèle

Pour indiquer quelle propriété de nœud contient des régressions étiquetées à des fins d'entraînement, ajoutez un élément de régression de nœud au tableau de cibles, en utilisant `"type" : "regression"`. Ajoutez un champ `node` pour indiquer le type de nœud des

nœuds cibles. Ajoutez un champ `predicate` pour définir les données littérales utilisées comme fonctionnalité de nœud cible du nœud cible. Ajoutez un champ `split_rate` si vous souhaitez remplacer le taux de division par défaut.

La cible `node` suivante indique que la propriété `rating` de chaque nœud `Movie` doit être traitée comme une étiquette de régression de nœud :

```
    "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/rating",
        "type": "regression",
        "split_rate": [0.7,0.1,0.2]
      }
    ]
  }
}
```

Spécification d'une tâche de prédiction de lien pour des arêtes spécifiques

Pour indiquer quelles arêtes doivent être utilisées à des fins d'entraînement de prédiction de lien, ajoutez un élément d'arête au tableau de cibles en utilisant `"type" : "link_prediction"`. Ajoutez les champs `subject`, `predicate` et `object` pour spécifier le type d'arête. Ajoutez un champ `split_rate` si vous souhaitez remplacer le taux de division par défaut.

La cible `edge` suivante indique que les arêtes `directed` qui connectent `Directors` à `Movies` doivent être utilisées pour la prédiction de lien :

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "subject": "http://aws.amazon.com/neptune/csv2rdf/class/Director",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/directed",
        "object": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "type" : "link_prediction"
      }
    ]
```

```
  }
}
```

Spécification d'une tâche de prédiction de lien pour toutes les arêtes

Pour indiquer que toutes les arêtes doivent être utilisées à des fins d'entraînement de prédiction de lien, ajoutez un élément `edge` au tableau de cibles en utilisant `"type" : "link_prediction"`. N'ajoutez pas de champs `subject`, `predicate` ni `object`. Ajoutez un champ `split_rate` si vous souhaitez remplacer le taux de division par défaut.

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "type" : "link_prediction"
      }
    ]
  }
}
```

Traitement des données de graphe exportées depuis Neptune pour l'entraînement

L'étape de traitement des données prend les données de graphe Neptune créées par le processus d'exportation et crée les informations utilisées par la bibliothèque Deep Graph Library (DGL) pendant l'entraînement. Cela inclut la réalisation de divers mappages et transformations de données :

- Analyse des nœuds et des arêtes pour créer les fichiers de mappage de graphes et d'identifiants requis par la bibliothèque DGL.
- Conversion des propriétés de nœud et d'arête en fonctionnalités de nœud et d'arête requises par la bibliothèque DGL.
- Division des données en jeux de données d'entraînement, de validation et de test.

Gestion de l'étape de traitement des données pour Neptune ML

Après avoir exporté les données de Neptune que vous souhaitez utiliser pour l'entraînement des modèles, vous pouvez démarrer une tâche de traitement des données à l'aide d'une commande comme celle-ci :

AWS CLI

```
aws neptunedata start-ml-data-processing-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --input-data-s3-location "s3://(S3 bucket name)/(path to your input folder)" \
  --id "(a job ID for the new job)" \
  --processed-data-s3-location "s3://(S3 bucket name)/(path to your output folder)"
 \
  --config-file-name "training-job-configuration.json"
```

Pour plus d'informations, consultez start-ml-data-processing-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
```

```
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_data_processing_job(
    inputDataS3Location='s3://(S3 bucket name)/(path to your input folder)',
    id='(a job ID for the new job)',
    processedDataS3Location='s3://(S3 bucket name)/(path to your output folder)',
    configFileName='training-job-configuration.json'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/dataprocessing \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(S3 bucket name)/(path to your input folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)",
        "configFileName" : "training-job-configuration.json"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(S3 bucket name)/(path to your input folder)",
        "id" : "(a job ID for the new job)",
```

```
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)",
        "configFileName" : "training-job-configuration.json"
      }'
```

Les détails de l'utilisation de cette commande sont expliqués dans Commande dataprocessing, avec d'autres informations sur la façon d'obtenir le statut d'une tâche en cours d'exécution, d'arrêter une tâche en cours d'exécution et de répertorier toutes les tâches en cours d'exécution.

Traitement de données de graphe mises à jour pour Neptune ML

Vous pouvez également fournir un `previousDataProcessingJobId` à l'API pour vous assurer que la nouvelle tâche de traitement de données utilise la même méthode de traitement qu'une tâche précédente. Cela est nécessaire lorsque vous souhaitez obtenir des prédictions pour des données de graphe mises à jour dans Neptune, soit en réentraînant l'ancien modèle sur les nouvelles données, soit en recalculant les artefacts de modèle sur les nouvelles données.

Pour ce faire, utilisez une commande comme celle-ci :

AWS CLI

```
aws neptunedata start-ml-data-processing-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --input-data-s3-location "s3://(Amazon S3 bucket name)/(path to your input
 folder)" \
  --id "(a job ID for the new job)" \
  --processed-data-s3-location "s3://(Amazon S3 bucket name)/(path to your output
 folder)" \
  --previous-data-processing-job-id "(the job ID of the previous data-processing
 job)"
```

Pour plus d'informations, consultez start-ml-data-processing-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
```

```
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_data_processing_job(
    inputDataS3Location='s3://(Amazon S3 bucket name)/(path to your input folder)',
    id='(a job ID for the new job)',
    processedDataS3Location='s3://(Amazon S3 bucket name)/(path to your output
 folder)',
    previousDataProcessingJobId='(the job ID of the previous data-processing job)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/dataprocessing \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your
 output folder)",
        "previousDataProcessingJobId" : "(the job ID of the previous data-processing
 job)"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/dataprocessing \
  -H 'Content-Type: application/json' \
```

```
 -d '{
       "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
folder)",
       "id" : "(a job ID for the new job)",
       "processedDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your
output folder)",
       "previousDataProcessingJobId" : "(the job ID of the previous data-processing
job)"
     }'
```

Définissez la valeur du paramètre `previousDataProcessingJobId` sur l'ID de tâche de la tâche de traitement de données précédente qui correspond au modèle entraîné.

> **Note**
>
> La suppression de nœud dans le graphe mis à jour n'est actuellement pas prise en charge. Si des nœuds ont été supprimés dans un graphe mis à jour, vous devez démarrer une toute nouvelle tâche de traitement de données plutôt qu'utiliser `previousDataProcessingJobId`.

Encodage des fonctionnalités dans Neptune ML

Les valeurs des propriétés se présentent sous différents formats et types de données. Pour obtenir de bonnes performances de machine learning, il est essentiel de convertir ces valeurs en encodages numériques appelés fonctionnalités.

Neptune ML effectue l'extraction et l'encodage des fonctionnalités dans le cadre des étapes d'exportation et de traitement des données, en utilisant les techniques d'encodage des fonctionnalités décrites ici.

> **Note**
>
> Si vous envisagez d'implémenter votre propre encodage de fonctionnalités dans une implémentation de modèle personnalisé, vous pouvez désactiver l'encodage automatique des fonctionnalités au cours de la phase de prétraitement des données en sélectionnant `none` comme type d'encodage des fonctionnalités. Aucun encodage des fonctionnalités n'est alors effectué sur ce nœud ou cette propriété d'arête, et les valeurs de propriété brutes sont analysées et enregistrées dans un dictionnaire. Le prétraitement des données crée toujours le graphe DGL à partir du jeu de données exporté, mais le graphe DGL construit ne possède pas les fonctionnalités prétraitées nécessaires à l'entraînement.
> Vous ne devez utiliser cette option que si vous envisagez d'effectuer l'encodage des fonctionnalités personnalisées dans le cadre de l'entraînement de modèle personnalisé. Consultez Modèles personnalisés dans Neptune ML pour plus de détails.

Fonctionnalités catégorielles dans Neptune ML

Une propriété qui peut prendre une ou plusieurs valeurs distinctes à partir d'une liste fixe de valeurs possibles est une fonctionnalité catégorielle. Dans Neptune ML, les fonctionnalités catégorielles sont encodées à l'aide d'un encodage 1 parmi n. L'exemple suivant montre comment le nom de propriété de différents aliments est encodé par encodage 1 parmi n en fonction de sa catégorie :

```
 Food        Veg.   Meat   Fruit    Encoding
---------    ----   ----   -----    --------
 Apple         0      0      1         001
 Chicken       0      1      0         010
 Broccoli      1      0      0         100
```

> **Note**
>
> Le nombre maximal de catégories par fonctionnalité catégorielle est de 100. Si une propriété comporte plus de 100 catégories de valeur, seules les 99 catégories les plus courantes sont placées dans des catégories distinctes, les autres étant placées dans une catégorie spéciale nommée `OTHER`.

Fonctionnalités numériques dans Neptune ML

Toute propriété dont les valeurs sont des nombres réels peut être encodée en tant que fonctionnalité numérique dans Neptune ML. Les fonctionnalités numériques sont encodées à l'aide de nombres à virgule flottante.

Vous pouvez spécifier une méthode de normalisation des données à utiliser lors de l'encodage de fonctionnalités numériques, comme ceci : `"norm": "normalization technique"`. Les techniques de normalisation suivantes sont prises en charge :

- « aucune » : ne normalisez pas les valeurs numériques pendant l'encodage.
- « min-max » : normalisez chaque valeur en lui soustrayant la valeur minimale, puis en la divisant par la différence entre la valeur maximale et la valeur minimale.
- « standard » : normalisez chaque valeur en la divisant par la somme de toutes les valeurs.

Fonctionnalités numériques de compartiment dans Neptune ML

Plutôt que de représenter une propriété numérique à l'aide de nombres bruts, vous pouvez condenser les valeurs numériques en catégories. Par exemple, vous pouvez diviser l'âge des personnes en catégories telles que les enfants (0 à 20), les jeunes adultes (20 à 40), les personnes d'âge moyen (40 à 60) et les personnes âgées (à partir de 60). En utilisant ces compartiments numériques, vous transformeriez une propriété numérique en une sorte de fonctionnalité catégorielle.

Dans Neptune ML, vous pouvez faire en sorte qu'une propriété numérique soit encodée sous la forme d'une fonctionnalité numérique de compartiment. Pour cela, vous devez fournir deux éléments :

- Une plage numérique sous la forme `"range": [a, b]`, où a et b sont des entiers.
- Nombre de compartiments, sous la forme `"bucket_cnt": c`, où c est le nombre de compartiments, également un entier.

Neptune ML calcule alors la taille de chaque compartiment sous la forme `( b - a ) / c` , et encode chaque valeur numérique en tant que numéro du compartiment dans lequel il se trouve. Toute valeur inférieure à a est considérée comme appartenant au premier compartiment, et toute valeur supérieure à b est considérée comme appartenant au dernier compartiment.

En option, vous pouvez également placer les valeurs numériques dans plusieurs compartiments, en spécifiant une taille de fenêtre coulissante, comme ceci : `"slide_window_size": s` , où `s` est un nombre. Neptune ML transforme ensuite chaque valeur numérique `v` de la propriété en une plage allant de `v - s/2` à `v + s/2` , et affecte la valeur `v` à chaque compartiment couvert par la plage.

Enfin, vous pouvez également éventuellement fournir un moyen de renseigner les valeurs manquantes pour les fonctionnalités numériques et les fonctionnalités numériques de compartiment. Pour ce faire, utilisez `"imputer": "imputation technique "`, où la technique d'imputation est l'une des suivantes : `"mean"`, `"median"` ou `"most-frequent"`. Si vous ne spécifiez aucun paramètre imputer, une valeur manquante peut entraîner l'arrêt du traitement.

Encodage des fonctionnalités de texte dans Neptune ML

Pour le texte de forme libre, Neptune ML peut utiliser plusieurs modèles différents pour convertir la séquence de jetons d'une chaîne de valeur de propriété en un vecteur de valeur réelle de taille fixe :

- `text_fasttext` : utilise l'encodage fastText. Il s'agit de l'encodage recommandé pour les fonctionnalités qui utilisent une et une seule des cinq langues prises en charge par fastText.
- `text_sbert` : utilise les modèles d'encodage Sentence BERT (SBERT). Il s'agit de l'encodage recommandé pour le texte que `text_fasttext` ne prend pas en charge.
- `text_word2vec` : utilise les algorithmes Word2Vec publiés à l'origine par Google pour encoder du texte. Word2Vec prend en charge uniquement l'anglais.
- `text_tfidf` : utilise un vectoriseur de fréquence de terme et de fréquence de document inverse (TF-IDF) pour l'encodage de texte. L'encodage TF-IDF prend en charge des fonctionnalités statistiques que les autres encodages ne prennent pas en charge.

Encodage fastText des valeurs de propriété de texte dans Neptune ML

Neptune ML peut utiliser les modèles fastText pour convertir les valeurs de propriété de texte en vecteurs de valeurs réelles de taille fixe. Il s'agit de la méthode d'encodage recommandée pour les valeurs de propriété de texte dans l'une des cinq langues prises en charge par fastText :

- `en` (anglais)
- `zh` (chinois)
- `hi` (hindi)
- `es` (espagnol)
- `fr` (français)

Notez que fastText ne peut pas traiter les phrases contenant des mots dans plusieurs langues.

La méthode `text_fasttext` peut éventuellement utiliser le champ `max_length` qui spécifie le nombre maximal de jetons dans une valeur de propriété de texte qui seront encodés, après quoi la chaîne est tronquée. Cela peut améliorer les performances lorsque les valeurs de propriété de texte contiennent de longues chaînes, car si `max_length` n'est pas spécifié, fastText encode tous les jetons, quelle que soit la longueur de la chaîne.

Cet exemple indique que les titres de films en français sont encodés à l'aide de fastText :

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
        "feature": ["title", "title", "text_fasttext"],
        "language": "fr",
        "max_length": 1024
      }
    ]
  }
```

Encodage Sentence BERT (SBERT) des fonctionnalités de texte dans Neptune ML

Neptune ML peut convertir la séquence de jetons d'une valeur de propriété de chaîne en un vecteur de valeur réelle de taille fixe à l'aide des modèles Sentence BERT (SBERT). Neptune prend en charge deux méthodes SBERT : `text_sbert128`, qui est la méthode par défaut si vous spécifiez simplement `text_sbert`, et `text_sbert512`. La différence entre les deux réside dans la longueur maximale d'une chaîne de valeur de propriété de texte qui est encodée. L'encodage `text_sbert128` tronque les chaînes de texte après l'encodage de 128 jetons, tandis que `text_sbert512` tronque les chaînes de texte après l'encodage de 512 jetons. Par conséquent,

`text_sbert512` nécessite plus de temps de traitement que `text_sbert128`. Les deux méthodes sont plus lentes que `text_fasttext`.

L'encodage SBERT étant multilingue, il n'est pas nécessaire de spécifier une langue pour le texte de la valeur de propriété que vous encodez. SBERT prend en charge de nombreuses langues et peut encoder une phrase contenant plusieurs langues. Si vous encodez des valeurs de propriété contenant du texte dans une ou plusieurs langues que fastText ne prend pas en charge, SBERT est la méthode d'encodage recommandée.

L'exemple suivant indique que les titres de films sont encodés au format SBERT jusqu'à un maximum de 128 jetons :

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      { "feature": ["title", "title", "text_sbert128"] }
    ]
  }
```

Encodage Word2Vec des fonctionnalités textuelles dans Neptune ML

Neptune ML peut encoder les valeurs de propriété de chaîne en tant que fonctionnalité Word2Vec (les algorithmes Word2Vec ont été initialement publiés par Google). La méthode `text_word2vec` encode les jetons d'une chaîne sous forme de vecteur dense à l'aide d'un des modèles entraînés par spaCy. Elle prend en charge uniquement la langue anglaise à l'aide du modèle en_core_web_lg.

L'exemple suivant indique que les titres des films sont encodés à l'aide de Word2Vec :

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
        "feature": ["title", "title", "text_word2vec"],
        "language": "en_core_web_lg"
      }
    ]
  }
```

Notez que le champ de langue est facultatif, car le modèle `en_core_web_lg` anglais est le seul pris en charge par Neptune.

Encodage TF-IDF des fonctionnalités de texte dans Neptune ML

Neptune ML peut encoder les valeurs de propriété de texte sous forme de fonctionnalités `text_tfidf`. Cet encodage convertit la séquence de mots du texte en un vecteur numérique à l'aide d'un vectoriseur de fréquence de terme et de fréquence de document inverse (TF-IDF), suivi d'une opération de réduction de dimensionnalité.

La TF-IDF (de fréquence de terme et de fréquence de document inverse) est une valeur numérique destinée à mesurer l'importance d'un mot dans un ensemble de documents. Elle est calculée en divisant le nombre de fois qu'un mot apparaît dans une valeur de propriété donnée par le nombre total de valeurs de propriété dans lesquelles il apparaît.

Par exemple, si le mot « kiss » apparaît deux fois dans un titre de film donné (par exemple, « kiss kiss bang bang ») et que « kiss » apparaît dans les titres de 4 films en tout, alors la valeur TF-IDF de « kiss » dans le titre « kiss kiss bang bang » est de `2 / 4` .

Le vecteur initialement créé possède d dimensions, où d est le nombre de termes uniques dans toutes les valeurs de propriété de ce type. L'opération de réduction de dimensionnalité utilise une projection fragmentée aléatoire pour réduire ce nombre à un maximum de 100. Le vocabulaire d'un graphe est ensuite généré en fusionnant toutes les fonctionnalités `text_tfidf` qu'il contient.

Vous pouvez contrôler le vectoriseur TF-IDF de plusieurs manières :

- **`max_features`** : le paramètre `max_features` vous permet de limiter le nombre de termes figurant dans les fonctionnalités `text_tfidf` aux termes les plus courants. Par exemple, si vous définissez `max_features` sur 100, seuls les 100 termes les plus couramment utilisés sont inclus. Si vous ne la définissez pas explicitement, la valeur par défaut de `max_features` est 5 000.
- **`min_df`** : le paramètre `min_df` vous permet de limiter le nombre de termes figurant dans les fonctionnalités `text_tfidf` à ceux ayant au moins une fréquence de document spécifiée. Par exemple, si vous définissez `min_df` sur 5, seuls les termes qui apparaissent dans au moins 5 valeurs de propriété différentes sont utilisés. Si vous ne la définissez pas explicitement, la valeur par défaut de `min_df` est 2.
- **`ngram_range`** : le paramètre `ngram_range` détermine quelles combinaisons de mots sont traitées comme des termes. Par exemple, si vous définissez `ngram_range` sur `[2, 4]`, les 6 termes suivants sont trouvés dans le titre « kiss kiss bang bang » :

- Termes de 2 mots : « kiss kiss », « kiss bang » et « bang bang ».
- Termes de 3 mots : « kiss kiss bang » et « kiss bang bang ».
- Termes de 4 mots : « kiss kiss bang bang ».

Le paramètre par défaut pour `ngram_range` est `[1, 1]`.

Fonctionnalités datetime dans Neptune ML

Neptune ML peut convertir des parties de valeurs de propriété `datetime` en fonctionnalités catégorielles en les encodant sous forme de tableaux d'encodage 1 parmi n. Utilisez le paramètre `datetime_parts` pour spécifier une ou plusieurs des parties suivantes à encoder : `["year", "month", "weekday", "hour"]`. Si vous ne définissez pas `datetime_parts`, les quatre parties sont encodées par défaut.

Par exemple, si la plage de valeurs datetime couvre les années 2010 à 2012, les quatre parties de l'entrée datetime `2011-04-22 01:16:34` sont les suivantes :

- year : `[0, 1, 0]`.

  Comme seulement 3 années sont couvertes (2010, 2011 et 2012), le tableau d'encodage 1 parmi n comporte trois entrées, une pour chaque année.
- month : `[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]`.

  Ici, le tableau d'encodage 1 parmi n comporte une entrée pour chaque mois de l'année.
- weekday : `[0, 0, 0, 0, 1, 0, 0]`.

  La norme ISO 8601 stipule que le lundi est le premier jour de la semaine, et comme le 22 avril 2011 était un vendredi, le tableau d'encodage 1 parmi n des jours de la semaine comporte un 1 en cinquième position.
- hour : `[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`.

  L'heure 1 h du matin est définie dans un tableau d'encodage 1 parmi n à 24 membres.

Le jour du mois, la minute et la seconde ne sont pas encodés de manière catégorielle.

Si la plage `datetime` totale en question inclut uniquement des dates d'une même année, aucun tableau `year` n'est encodé.

Vous pouvez définir une stratégie d'imputation pour compléter les valeurs `datetime` manquantes, à l'aide du paramètre `imputer` et de l'une des stratégies disponibles pour les fonctionnalités numériques.

Encodage automatique des fonctionnalités dans Neptune ML

Au lieu de spécifier manuellement les méthodes d'encodage de fonctionnalité à utiliser pour les propriétés figurant dans votre graphe, vous pouvez définir `auto` en tant que méthode d'encodage de fonctionnalité. Neptune ML tente alors de déduire le meilleur encodage de fonctionnalité pour chaque propriété en fonction de son type de données sous-jacent.

Voici quelques-unes des heuristiques utilisées par Neptune ML pour sélectionner les encodages de fonctionnalité appropriés :

- Si la propriété comporte uniquement des valeurs numériques et peut être convertie en types de données numériques, Neptune ML l'encode généralement sous forme de valeur numérique. Toutefois, si le nombre de valeurs uniques pour la propriété est inférieur à 10 % du nombre total de valeurs et que la cardinalité de ces valeurs uniques est inférieure à 100, Neptune ML utilise un encodage catégoriel.
- Si les valeurs de propriété peuvent être converties en un type `datetime`, Neptune ML les encode en tant que fonctionnalité `datetime`.
- Si les valeurs de propriété peuvent être converties de manière forcée en booléens (1/0 ou True/False), Neptune ML utilise un encodage de catégorie.
- Si la propriété est une chaîne dont plus de 10 % de ses valeurs sont uniques et que le nombre moyen de jetons par valeur est supérieur ou égal à 3, Neptune ML en déduit que le type de propriété est texte et détecte automatiquement la langue utilisée. Si la langue détectée est l'une de celles prises en charge par fastText, à savoir l'anglais, le chinois, l'hindi, l'espagnol ou le français, Neptune ML utilise `text_fasttext` pour encoder le texte. Dans le cas contraire, Neptune ML utilise `text_sbert`.
- Si la propriété est une chaîne non classée en tant que fonctionnalité de texte, Neptune ML suppose qu'il s'agit d'une fonctionnalité catégorielle et utilise un encodage de catégorie.
- Si chaque nœud possède sa propre valeur unique pour une propriété qui est déduite comme étant une fonctionnalité de catégorie, Neptune ML supprime la propriété du graphe d'entraînement car il s'agit probablement d'un identifiant sans valeur informative pour l'apprentissage.
- Si la propriété est connue pour contenir des séparateurs Neptune valides tels que des points-virgules (« ; »), Neptune ML peut uniquement traiter la propriété en tant que `MultiNumerical` ou `MultiCategorical`.

  - Neptune ML essaie d'abord d'encoder les valeurs sous forme de fonctionnalités numériques. En cas de succès, Neptune ML utilise l'encodage numérique pour créer des fonctionnalités vectorielles numériques.
  - Dans le cas contraire, Neptune ML les encode en tant que valeurs multi-catégorielles.
- Si Neptune ML ne peut pas déduire le type de données des valeurs d'une propriété, Neptune est la propriété à partir MLdrops du graphe d'apprentissage.

Modification d'un fichier de configuration de données d'entraînement

Le processus d'exportation Neptune exporte les données Neptune ML à partir d'un cluster de bases de données Neptune vers un compartiment S3. Il exporte les nœuds et les arêtes séparément dans un dossier `nodes/` et un dossier `edges/`. Il crée également un fichier de configuration de données d'entraînement JSON, nommé `training-data-configuration.json` par défaut. Ce fichier contient des informations sur le schéma du graphe, les types de ses fonctionnalités, les opérations de transformation et de normalisation de fonctionnalités, ainsi que la fonctionnalité cible pour une tâche de classification ou de régression.

Il peut arriver que vous souhaitiez modifier directement le fichier de configuration. C'est notamment le cas lorsque vous souhaitez modifier la façon dont les fonctionnalités sont traitées ou la façon dont le graphe est construit, sans avoir à réexécuter l'exportation chaque fois que vous souhaitez modifier la spécification de la tâche de machine learning que vous êtes en train de résoudre.

Pour modifier le fichier de configuration de données d'entraînement

1. Téléchargez le fichier sur votre ordinateur local.

   À moins que vous n'ayez spécifié une ou plusieurs tâches nommées dans le paramètre `additionalParams/neptune_ml` transmis au processus d'exportation, le fichier portera le nom par défaut, qui est `training-data-configuration.json`. Vous pouvez utiliser une commande AWS CLI comme celle-ci pour télécharger le fichier :

   ```
   aws s3 cp \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json \
     ./
   ```

2. Modifiez le fichier à l'aide d'un éditeur de texte.
3. Chargez le fichier modifié. Chargez le fichier modifié au même emplacement dans Amazon S3 à partir duquel vous l'avez téléchargé, à l'aide d'une commande AWS CLI comme celle-ci :

   ```
   aws s3 cp \
     training-data-configuration.json \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json
   ```

Exemple de fichier de configuration de données d'entraînement JSON

Voici un exemple de fichier de configuration des données d'entraînement qui décrit un graphe pour une tâche de classification de nœud :

```
{
  "version" : "v2.0",
  "query_engine" : "gremlin",
  "graph" : [
    {
      "edges" : [
        {
          "file_name" : "edges/(movie)-included_in-(genre).csv",
          "separator" : ",",
          "source" : ["~from", "movie"],
          "relation" : ["", "included_in"],
          "dest" : [ "~to", "genre" ]
        },
        {
          "file_name" : "edges/(user)-rated-(movie).csv",
          "separator" : ",",
          "source" : ["~from", "movie"],
          "relation" : ["rating", "prefixname"], # [prefixname#value]
          "dest" : ["~to", "genre"],
          "features" : [
            {
              "feature" : ["rating", "rating", "numerical"],
              "norm" : "min-max"
            }
          ]
        }
      ],
      "nodes" : [
        {
          "file_name" : "nodes/genre.csv",
          "separator" : ",",
          "node" : ["~id", "genre"],
          "features" : [
            {
              "feature": ["name", "genre", "category"],
              "separator": ";"
            }
          ]
```

```
        },
        {
          "file_name" : "nodes/movie.csv",
          "separator" : ",",
          "node" : ["~id", "movie"],
          "features" : [
            {
              "feature": ["title", "title", "word2vec"],
              "language": ["en_core_web_lg"]
            }
          ]
        },
        {
          "file_name" : "nodes/user.csv",
          "separator" : ",",
          "node" : ["~id", "user"],
          "features" : [
            {
              "feature": ["age", "age", "numerical"],
              "norm" : "min-max",
              "imputation": "median",
            },
            {
              "feature": ["occupation", "occupation", "category"],
            }
          ],
          "labels" : [
            {
              "label": ["gender", "classification"],
              "split_rate" : [0.8, 0.2, 0.0]
            }
          ]
        }
      ]
    },
    "warnings" : [ ]
  ]
}
```

Structure des fichiers de configuration des données d'entraînement JSON

Le fichier de configuration d'entraînement fait référence aux fichiers CSV enregistrés par le processus d'exportation dans les dossiers `nodes/` et `edges/`.

Chaque fichier situé sous `nodes/` stocke des informations sur les nœuds qui ont la même étiquette de nœud de graphe de propriétés. Chaque colonne d'un fichier de nœud stocke soit l'ID de nœud, soit la propriété de nœud. La première ligne du fichier contient un en-tête qui spécifie l'élément `~id` ou le nom de propriété pour chaque colonne.

Chaque fichier situé sous `edges/` stocke des informations sur les nœuds dotés de la même étiquette d'arête de graphe de propriétés. Chaque colonne d'un fichier de nœud stocke l'ID de nœud source, l'ID de nœud de destination ou la propriété d'arête. La première ligne du fichier contient un en-tête qui spécifie l'élément `~from`, l'élément `~to` ou le nom de propriété pour chaque colonne.

Le fichier de configuration de données d'entraînement comporte trois éléments de niveau supérieur :

```
{
  "version" : "v2.0",
  "query_engine" : "gremlin",
  "graph" : [ ... ]
}
```

- `version` : (string) version utilisée du fichier de configuration.
- `query_engine` (string) langage de requête utilisé pour l'exportation des données de graphe. Pour le moment, seul « gremlin » est valide.
- `graph` : (tableau JSON) répertorie un ou plusieurs objets de configuration contenant les paramètres de modèle pour chacun des nœuds et des arêtes qui seront utilisés.

  Les objets de configuration du tableau de graphe ont la structure décrite dans la section suivante.

Contenu d'un objet de configuration répertorié dans le tableau **`graph`**

Un objet de configuration figurant dans le tableau `graph` peut contenir trois nœuds de niveau supérieur :

```
    {
      "edges"    : [ ... ],
      "nodes"    : [ ... ],
      "warnings" : [ ... ],
    }
```

- `edges` : (tableau d'objets JSON) chaque objet JSON spécifie un ensemble de paramètres pour définir la manière dont une arête du graphe sera traitée pendant le traitement et l'entraînement de modèle. Ceci n'est utilisé qu'avec le moteur Gremlin.
- `nodes` : (tableau d'objets JSON) chaque objet JSON spécifie un ensemble de paramètres pour définir la manière dont un nœud du graphe sera traité pendant le traitement et l'entraînement de modèle. Ceci n'est utilisé qu'avec le moteur Gremlin.
- `warnings` : (tableau d'objets JSON) chaque objet contient un avertissement généré pendant le processus d'exportation de données.

Contenu d'un objet de configuration d'arête répertorié dans un tableau **`edges`**

Un objet de configuration d'arête répertorié dans un tableau `edges` peut contenir les champs de niveau supérieur suivants :

```
    {
      "file_name" : "(path to a CSV file)",
      "separator" : "(separator character)",
      "source"    : ["(column label for starting node ID)", "(starting node type)"],
      "relation"  : ["(column label for the relationship name)", "(the prefix name
for the relationship name)"],
      "dest"      : ["(column label for ending node ID)", "(ending node type)"],
      "features"  : [(array of feature objects)],
      "labels"    : [(array of label objects)]
    }
```

- **`file_name`** : chaîne spécifiant le chemin d'un fichier CSV qui stocke des informations sur les arêtes ayant la même étiquette de graphe de propriétés.

  La première ligne de ce fichier contient une ligne d'en-tête d'étiquettes de colonne.

  Les deux premières étiquettes de colonne sont `~from` et `~to`. La première colonne (la colonne `~from`) stocke l'ID du nœud de départ de l'arête, et la seconde (la colonne `~to`) stocke l'ID du nœud de fin de l'arête.

  Les étiquettes de colonne restantes dans la ligne d'en-tête spécifient, pour chaque colonne restante, le nom de la propriété d'arête dont les valeurs ont été exportées dans cette colonne.
- **`separator`** : chaîne contenant le délimiteur qui sépare les colonnes dans ce fichier CSV.

- **`source`** : tableau JSON contenant deux chaînes qui spécifient le nœud de départ de l'arête. La première chaîne contient le nom d'en-tête de la colonne dans laquelle l'ID de nœud de départ est stocké. La deuxième chaîne spécifie le type de nœud.
- **`relation`** : tableau JSON contenant deux chaînes qui spécifient le type de relation de l'arête. La première chaîne contient le nom d'en-tête de la colonne dans laquelle le nom de relation (`relname`) est stocké. La deuxième chaîne contient le préfixe du nom de relation (`prefixname`).

  Le type de relation complet se compose des deux chaînes combinées, avec un trait d'union entre elles, comme ceci : *`prefixname`*`-`*`relname`*.

  Si la première chaîne est vide, toutes les arêtes ont le même type de relation, à savoir la chaîne `prefixname`.
- **`dest`** : tableau JSON contenant deux chaînes qui spécifient le nœud de fin de l'arête. La première chaîne contient le nom d'en-tête de la colonne dans laquelle l'ID de nœud est stocké. La deuxième chaîne spécifie le type de nœud.
- **`features`** : tableau JSON d'objets de fonctionnalité à valeur de propriété. Chaque objet de fonctionnalité à valeur de propriété contient les champs suivants :
  - feature : tableau JSON de trois chaînes. La première chaîne contient le nom d'en-tête de la colonne qui contient la valeur de propriété. La deuxième chaîne contient le nom de la fonctionnalité. La troisième chaîne contient le type de fonctionnalité.
  - norm : (facultatif) spécifie une méthode de normalisation à appliquer aux valeurs de propriété.

- **`labels`** : tableau JSON d'objets. Chacun des objets définit une fonctionnalité cible des arêtes et spécifie les proportions des arêtes que doivent prendre les phases d'entraînement et de validation. Chaque objet contient les champs suivants :
  - label : tableau JSON de deux chaînes. La première chaîne contient le nom d'en-tête de la colonne qui contient la valeur de propriété de la fonctionnalité cible. La deuxième chaîne spécifie l'un des types de tâches cibles suivants :
    - `"classification"` : tâche de classification d'arête. Les valeurs de propriété fournies dans la colonne identifiée par la première chaîne du tableau `label` sont traitées comme des valeurs catégorielles. Pour une tâche de classification d'arête, la première chaîne du tableau `label` ne peut pas être vide.
    - `"regression"` : tâche de régression d'arête. Les valeurs de propriété fournies dans la colonne identifiée par la première chaîne du tableau `label` sont traitées comme des valeurs numériques. Pour une tâche de régression d'arête, la première chaîne du tableau `label` ne peut pas être vide.

  - `"link_prediction"` : tâche de prédiction de lien. Aucune valeur de propriété n'est requise. Pour une tâche de prédiction de lien, la première chaîne du tableau `label` est ignorée.
- **`split_rate`** : tableau JSON contenant trois nombres compris entre zéro et un dont la somme est égale à un et qui représentent une estimation des proportions de nœuds que les phases d'entraînement, de validation et de test utiliseront, respectivement. Ce champ ou l'élément `custom_split_filenames` peut être défini, mais pas les deux. Consultez split_rate.
- **`custom_split_filenames`** : objet JSON qui spécifie les noms des fichiers qui définissent les populations d'entraînement, de validation et de test. Ce champ ou l'élément `split_rate` peut être défini, mais pas les deux. Pour plus d'informations, consultez train-validation-testProportions personnalisées.

Contenu d'un objet de configuration de nœud répertorié dans un tableau **`nodes`**

Un objet de configuration de nœud répertorié dans un tableau `nodes` peut contenir les champs suivants :

```
    {
      "file_name" : "(path to a CSV file)",
      "separator" : "(separator character)",
      "node"      : ["(column label for the node ID)", "(node type)"],
      "features"  : [(feature array)],
      "labels"    : [(label array)],
    }
```

- **`file_name`** : chaîne spécifiant le chemin d'un fichier CSV qui stocke des informations sur les nœuds ayant la même étiquette de graphe de propriétés.

  La première ligne de ce fichier contient une ligne d'en-tête d'étiquettes de colonne.

  La première étiquette de colonne est `~id` et la première colonne (la colonne `~id`) stocke l'ID de nœud.

  Les étiquettes de colonne restantes dans la ligne d'en-tête spécifient, pour chaque colonne restante, le nom de la propriété de nœud dont les valeurs ont été exportées dans cette colonne.
- **`separator`** : chaîne contenant le délimiteur qui sépare les colonnes dans ce fichier CSV.
- **`node`** : tableau JSON contenant deux chaînes. La première chaîne contient le nom d'en-tête de la colonne qui stocke le nœud IDs. La deuxième chaîne spécifie le type de nœud dans le graphe, qui correspond à une étiquette de graphe de propriétés du nœud.

- **`features`** : tableau JSON d'objets de fonctionnalité de nœud. Consultez Contenu d'un objet de fonctionnalité répertorié dans un tableau `features` pour un nœud ou une arête.
- **`labels`** : tableau JSON d'objets d'étiquette de nœud. Consultez Contenu d'un objet d'étiquette de nœud répertorié dans un tableau `labels` de nœud.

Contenu d'un objet de fonctionnalité répertorié dans un tableau **`features`** pour un nœud ou une arête

Un objet de fonctionnalité de nœud répertorié dans un tableau `features` de nœud peut contenir les champs supérieurs suivants :

- **`feature`** : tableau JSON de trois chaînes. La première chaîne contient le nom d'en-tête de la colonne qui contient la valeur de propriété de la fonctionnalité. La deuxième chaîne contient le nom de la fonctionnalité.

  La troisième chaîne contient le type de fonctionnalité. Les types de fonctionnalité valides sont répertoriés dans Valeurs possibles du champ de type pour les entités.
- **`norm`** : ce champ est obligatoire pour les fonctionnalités numériques. Il spécifie une méthode de normalisation à utiliser sur les valeurs numériques. Les valeurs valides sont `"none"`, `"min-max"` et « standard ». Consultez Le champ des normes pour plus de détails.
- **`language`** : le champ de langue indique la langue utilisée dans les valeurs des propriétés de texte. Son utilisation dépend de la méthode d'encodage de texte :
  - Pour l'encodage `text_fasttext`, ce champ est obligatoire et doit spécifier l'une des langues suivantes :
    - `en` (anglais)
    - `zh` (chinois)
    - `hi` (hindi)
    - `es` (espagnol)
    - `fr` (français)

    Toutefois, `text_fasttext` ne peut pas gérer plus d'une langue à la fois.
  - Pour l'encodage `text_sbert`, ce champ n'est pas utilisé, car l'encodage SBERT est multilingue.

  - Pour l'encodage `text_word2vec`, ce champ est facultatif, car `text_word2vec` prend en charge uniquement l'anglais. Le cas échéant, il doit spécifier le nom du modèle de langue anglaise :

    ```
    "language" : "en_core_web_lg"
    ```

  - Pour l'encodage `tfidf`, ce champ n'est pas utilisé.

- **`max_length`** : ce champ est facultatif pour les fonctionnalités `text_fasttext`, où il indique le nombre maximal de jetons d'une fonctionnalité de teste d'entrée qui seront encodés. Le texte d'entrée une fois la longueur `max_length` atteinte est ignoré. Par exemple, définir max_length sur 128 indique que tous les jetons situés après le 128e dans une séquence de texte sont ignorés.
- **`separator`** : ce champ est utilisé en option avec les fonctionnalités `category`, `numerical` et `auto`. Il spécifie un caractère qui peut être utilisé pour diviser une valeur de propriété en plusieurs valeurs catégorielles ou valeurs numériques.

  Consultez Le champ du séparateur.
- **`range`** : ce champ est obligatoire pour les fonctionnalités `bucket_numerical`. Il indique la plage de valeurs numériques qui doit être divisée en compartiments.

  Consultez Le champ de portée.
- **`bucket_cnt`** : ce champ est obligatoire pour les fonctionnalités `bucket_numerical`. Il spécifie le nombre de compartiments dans lesquels la plage numérique définie par le paramètre `range` doit être divisée.

  Consultez Fonctionnalités numériques de compartiment dans Neptune ML.
- **`slide_window_size`** : ce champ est utilisé en option avec les fonctionnalités `bucket_numerical` pour attribuer des valeurs à plusieurs compartiments.

  Consultez Le champ slide_window_size.
- **`imputer`** : ce champ est utilisé en option avec les fonctionnalités `numerical`, `bucket_numerical` et `datetime` pour fournir une technique d'imputation permettant de renseigner les valeurs manquantes. Les techniques d'imputation prises en charge sont `"mean"`, `"median"` et `"most_frequent"`.

  Consultez Le champ de l'imputeur.
- **`max_features`** : ce champ est utilisé en option par les fonctionnalités `text_tfidf` pour spécifier le nombre maximal de termes à encoder.

Consultez Le champ max_features.

- **`min_df`** : ce champ est utilisé en option par les fonctionnalités `text_tfidf` pour spécifier la fréquence minimale de document des termes à encoder.

  Consultez Le champ min_df.

- **`ngram_range`** : ce champ est utilisé en option par les fonctionnalités `text_tfidf` pour spécifier une plage de nombres de mots ou de jetons à considérer comme des termes individuels potentiels à encoder.

  Consultez Le champ ngram_range.

- **`datetime_parts`** : ce champ est utilisé en option par les fonctionnalités `datetime` pour spécifier les parties de la valeur datetime à encoder de manière catégorielle.

  Consultez Le champ datetime_parts.

Contenu d'un objet d'étiquette de nœud répertorié dans un tableau **`labels`** de nœud

Un objet d'étiquette répertorié dans un tableau `labels` de nœud définit une fonctionnalité cible de nœud et spécifie les proportions de nœuds que les phases d'entraînement, de validation et de test utiliseront. Chaque objet peut contenir les champs suivants :

```
     {
       "label"      : ["(column label for the target feature property value)", "(task
type)"],
       "split_rate" : [(training proportion), (validation proportion), (test
proportion)],
       "custom_split_filenames" : {"train": "(training file name)", "valid":
"(validation file name)", "test": "(test file name)"},
       "separator"  : "(separator character for node-classification category values)",
     }
```

- **`label`** : tableau JSON contenant deux chaînes. La première chaîne contient le nom d'en-tête de la colonne qui stocke les valeurs de propriété de la fonctionnalité. La deuxième chaîne indique le type de tâche cible, qui peut être :
  - `"classification"` : une tâche de classification de nœud. Les valeurs de propriété de la colonne spécifiée sont utilisées pour créer une fonctionnalité catégorielle.

  - `"regression"` : une tâche de régression de nœud. Les valeurs de propriété de la colonne spécifiée sont utilisées pour créer une fonctionnalité numérique.
- **`split_rate`** : tableau JSON contenant trois nombres compris entre zéro et un dont la somme est égale à un et qui représentent une estimation des proportions de nœuds que les phases d'entraînement, de validation et de test utiliseront, respectivement. Consultez split_rate.
- **`custom_split_filenames`** : objet JSON qui spécifie les noms des fichiers qui définissent les populations d'entraînement, de validation et de test. Ce champ ou l'élément `split_rate` peut être défini, mais pas les deux. Pour plus d'informations, consultez train-validation-testProportions personnalisées.
- **`separator`** : chaîne contenant le délimiteur qui sépare les valeurs des fonctionnalités catégorielles pour une tâche de classification.

> **Note**
>
> Si aucun objet d'étiquette n'est fourni à la fois pour les arêtes et les nœuds, il est automatiquement supposé que la tâche est une prédiction de lien et les arêtes sont réparties de façon aléatoire à 90 % pour l'entraînement et à 10 % pour la validation.

train-validation-testProportions personnalisées

Par défaut, le paramètre `split_rate` est utilisé par Neptune ML pour diviser le graphe de manière aléatoire en populations d'entraînement, de validation et de test en utilisant les proportions définies dans ce paramètre. Pour avoir un contrôle plus précis sur les entités utilisées dans ces différentes populations, il est possible de créer des fichiers qui les définissent explicitement, puis de modifier le fichier de configuration des données d'entraînement pour mapper ces fichiers d'indexation aux populations. Ce mappage est spécifié par un objet JSON pour la clé `custom_split_filesnames` dans le fichier de configuration d'entraînement. Si cette option est utilisée, les noms de fichiers doivent être fournis pour les clés `train` et `validation`, et sont facultatifs pour la clé `test`.

Le formatage de ces fichiers doit correspondre au format de données Gremlin. Plus précisément, pour les tâches au niveau du nœud, chaque fichier doit contenir une colonne avec l'`~id`en-tête répertoriant le nœud IDs, et pour les tâches au niveau de la périphérie, les fichiers doivent spécifier `~from` et `~to` indiquer les nœuds source et de destination des bords, respectivement. Ces fichiers doivent être placés dans le même emplacement Amazon S3 que les données exportées utilisées pour le traitement des données (voir : `outputS3Path`).

Pour les tâches de classification ou de régression de propriété, ces fichiers peuvent éventuellement définir les étiquettes de la tâche de machine learning. Dans ce cas, les fichiers doivent comporter une colonne de propriété portant le même nom d'en-tête que celui défini dans le fichier de configuration des données d'entraînement. Si des étiquettes de propriété sont définies à la fois dans les fichiers de nœud et d'arête exportés et dans les fichiers de division personnalisée, la priorité est donnée aux fichiers de division personnalisée.

Entraînement d'un modèle à l'aide de Neptune ML

Après avoir traité les données que vous avez exportées depuis Neptune pour l'entraînement des modèles, vous pouvez démarrer une tâche d'entraînement des modèles à l'aide d'une commande telle que la suivante :

AWS CLI

```
aws neptunedata start-ml-model-training-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique model-training job ID)" \
  --data-processing-job-id "(the data-processing job-id of a completed job)" \
  --train-model-s3-location "s3://(your S3 bucket)/neptune-model-graph-autotrainer"
```

Pour plus d'informations, consultez start-ml-model-training-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_model_training_job(
    id='(a unique model-training job ID)',
    dataProcessingJobId='(the data-processing job-id of a completed job)',
    trainModelS3Location='s3://(your S3 bucket)/neptune-model-graph-autotrainer'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltraining \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
```

```
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your S3 bucket)/neptune-model-graph-
autotrainer"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/modeltraining \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your S3 bucket)/neptune-model-graph-
autotrainer"
      }'
```

Les détails de l'utilisation de cette commande sont expliqués dans Commande modeltraining, avec d'autres informations sur la façon d'obtenir le statut d'une tâche en cours d'exécution, d'arrêter une tâche en cours d'exécution et de répertorier toutes les tâches en cours d'exécution.

Vous pouvez également fournir un identifiant `previousModelTrainingJobId` pour utiliser les informations issues d'une tâche d'entraînement de modèle Neptune ML terminée, afin d'accélérer la recherche d'hyperparamètres dans une nouvelle tâche d'entraînement. Cela est utile lors du réentraînement de modèle sur de nouvelles données de graphe, ainsi que lors de l'entraînement incrémentiel sur les mêmes données de graphe. Utilisez une commande comme celle-ci :

AWS CLI

```
aws neptunedata start-ml-model-training-job \
  --endpoint-url https://your-neptune-endpoint:port \
```

```
  --id "(a unique model-training job ID)" \
  --data-processing-job-id "(the data-processing job-id of a completed job)" \
  --train-model-s3-location "s3://(your S3 bucket)/neptune-model-graph-autotrainer"
 \
  --previous-model-training-job-id "(the model-training job-id of a completed job)"
```

Pour plus d'informations, consultez start-ml-model-training-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_model_training_job(
    id='(a unique model-training job ID)',
    dataProcessingJobId='(the data-processing job-id of a completed job)',
    trainModelS3Location='s3://(your S3 bucket)/neptune-model-graph-autotrainer',
    previousModelTrainingJobId='(the model-training job-id of a completed job)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltraining \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your S3 bucket)/neptune-model-graph-
autotrainer",
        "previousModelTrainingJobId" : "(the model-training job-id of a completed
 job)"
```

```
    }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/modeltraining \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your S3 bucket)/neptune-model-graph-
autotrainer",
        "previousModelTrainingJobId" : "(the model-training job-id of a completed
 job)"
      }'
```

Vous pouvez entraîner votre propre implémentation de modèle sur l'infrastructure d'entraînement Neptune ML en fournissant un objet `customModelTrainingParameters` tel que celui-ci :

AWS CLI

```
aws neptunedata start-ml-model-training-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique model-training job ID)" \
  --data-processing-job-id "(the data-processing job-id of a completed job)" \
  --train-model-s3-location "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer" \
  --model-name "custom" \
  --custom-model-training-parameters '{
    "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
    "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
    "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
```

```
  }'
```

Pour plus d'informations, consultez start-ml-model-training-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_model_training_job(
    id='(a unique model-training job ID)',
    dataProcessingJobId='(the data-processing job-id of a completed job)',
    trainModelS3Location='s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer',
    modelName='custom',
    customModelTrainingParameters={
        'sourceS3DirectoryPath': 's3://(your Amazon S3 bucket)/(path to your Python
 module)',
        'trainingEntryPointScript': '(your training script entry-point name in the
 Python module)',
        'transformEntryPointScript': '(your transform script entry-point name in the
 Python module)'
    }
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltraining \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
```

```
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer",
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your
 Python module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in
 the Python module)"
        }
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/modeltraining \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer",
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your
 Python module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in
 the Python module)"
        }
      }'
```

Consultez Commande modeltraining pour plus d'informations, notamment sur la façon d'obtenir le statut d'une tâche en cours d'exécution, la façon d'arrêter une tâche en cours d'exécution et la façon de répertorier toutes les tâches en cours d'exécution. Consultez Modèles personnalisés dans Neptune ML pour en savoir plus sur la façon d'implémenter et d'utiliser un modèle personnalisé.

Rubriques

- Modèles et entraînement de modèle dans Amazon Neptune ML
- Personnalisation des configurations d'hyperparamètres de modèle dans Neptune ML
- Bonnes pratiques d'entraînement de modèle

Modèles et entraînement de modèle dans Amazon Neptune ML

Neptune ML utilise les réseaux de neurones en graphes (GNN) pour créer des modèles pour les différentes tâches de machine learning. Il a été démontré que les réseaux neuronaux graphiques obtiennent des state-of-the-art résultats pour les tâches d'apprentissage automatique des graphes et sont excellents pour extraire des modèles informatifs à partir de données structurées par des graphes.

Réseaux neuronaux graphiques (GNNs) dans Neptune ML

Les réseaux neuronaux à graphes (GNNs) appartiennent à une famille de réseaux neuronaux qui calculent les représentations des nœuds en tenant compte de la structure et des caractéristiques des nœuds voisins. GNNs complètent d'autres méthodes traditionnelles d'apprentissage automatique et de réseaux neuronaux qui ne sont pas bien adaptées aux données graphiques.

GNNs sont utilisés pour résoudre des tâches d'apprentissage automatique telles que la classification et la régression des nœuds (prédiction des propriétés des nœuds), la classification et la régression des arêtes (prédiction des propriétés des arêtes) ou la prédiction des liens (prédiction si deux nœuds du graphe doivent être connectés ou non).

En général, l'utilisation d'un réseau GNN pour une tâche de machine learning comporte deux phases :

- Une phase d'encodage, au cours de laquelle le réseau GNN calcule un vecteur à d dimensions pour chaque nœud du graphe. Ces vecteurs sont également appelés représentations ou intégrations.
- Une phase de décodage, qui effectue des prédictions basées sur les représentations encodées.

Pour la classification et la régression de nœud, les représentations des nœuds sont utilisées directement pour les tâches de classification et de régression. Pour la classification et la régression d'arête, les représentations de nœud des nœuds incidents sur une arête sont utilisées comme entrée pour la classification ou la régression. Pour la prédiction de lien, un score de probabilité d'arête est calculé à l'aide d'une paire de représentations de nœud et d'une représentation de type d'arête.

La Deep Graph Library (DGL) facilite la définition et la formation efficaces GNNs de ces tâches.

Différents modèles GNN sont unifiés dans le cadre de la formulation de la transmission de messages. Dans cette vue, la représentation d'un nœud dans un graphe est calculée à l'aide des représentations

des voisins du nœud (les messages), ainsi que de la représentation initiale du nœud. Dans NeptuneML, la représentation initiale d'un nœud est dérivée des fonctionnalités extraites de ses propriétés de nœud, ou elle peut être apprise et dépend de l'identité du nœud.

Neptune ML offre également la possibilité de concaténer les fonctionnalités de nœud et les représentations de nœud pouvant être apprises pour servir de représentation de nœud d'origine.

Pour les diverses tâches dans Neptune ML impliquant des graphes dotés de propriétés de nœud, nous utilisons le réseau convolutif graphique relationnel (R-GCN) pour effectuer la phase d'encodage. R-GCN est une architecture GNN parfaitement adaptée aux graphes comportant plusieurs types de nœud et d'arête (ils sont appelés graphes hétérogènes).

Le réseau R-GCN est constitué d'un nombre fixe de couches empilées les unes après les autres. Chaque couche du R-GCN utilise ses paramètres de modèle pouvant être appris pour agréger les informations provenant du voisinage à un saut, immédiat d'un nœud. Comme les couches suivantes utilisent les représentations en sortie de la couche précédente comme entrée, le rayon du voisinage de graphe qui influence l'intégration finale d'un nœud dépend du nombre de couches (`num-layer`) du réseau R-GCN.

Par exemple, cela signifie qu'un réseau à deux couches utilise les informations provenant de nœuds situés à deux sauts de distance.

Pour en savoir plus GNNs, consultez A Comprehensive Survey on Graph Neural Networks. Pour plus d'informations sur la bibliothèqe Deep Graph Library (DGL), visitez la page Web de la DGL. Pour un didacticiel pratique sur l'utilisation de DGL avec GNNs, voir Apprendre les réseaux neuronaux à graphes avec Deep Graph Library.

Réseaux de neurones en graphes d'entraînement

Dans le machine learning, le processus qui consiste à amener un modèle à apprendre à effectuer de bonnes prédictions pour une tâche s'appelle « entraînement de modèle ». Il est généralement réalisé en spécifiant un objectif particulier à optimiser, ainsi qu'un algorithme à utiliser pour effectuer cette optimisation.

Ce processus est utilisé pour entraîner un réseau GNN à apprendre de bonnes représentations également pour la tâche en aval. Nous créons une fonction objectif pour cette tâche qui est minimisée lors de l'entraînement du modèle. Par exemple, pour la classification des nœuds, nous utilisons CrossEntropyLosscomme objectif, ce qui pénalise les erreurs de classification, et pour la régression des nœuds, nous minimisons. MeanSquareError

L'objectif est généralement une fonction de perte qui prend les prédictions du modèle pour un point de données particulier et les compare à la valeur réelle pour ce point de données. Il renvoie la valeur de perte, qui indique à quel point les prédictions du modèle sont éloignées. L'objectif du processus d'entraînement est de minimiser les pertes et de s'assurer que les prédictions du modèle sont proches de la réalité.

L'algorithme d'optimisation utilisé en deep learning pour le processus d'entraînement est généralement une variante de la descente de gradient. Dans Neptune ML, nous utilisons Adam, un algorithme d'optimisation des fonctions objectives stochastiques basé sur un gradient de premier ordre, sur la base d'estimations adaptatives de moments d'ordre inférieur.

Alors que le processus d'entraînement de modèle essaie de s'assurer que les paramètres de modèle appris sont proches des minima de la fonction objectif, les performances globales d'un modèle dépendent également des hyperparamètres du modèle, qui sont des paramètres de modèle qui ne sont pas appris par l'algorithme d'entraînement. Par exemple, la dimensionnalité de la représentation de nœud appris, `num-hidden`, est un hyperparamètre qui affecte les performances du modèle. Par conséquent, il est courant en machine learning d'effectuer une optimisation des hyperparamètres (HPO) pour choisir les hyperparamètres appropriés.

Neptune ML utilise une tâche de réglage des hyperparamètres basée sur l' SageMaker IA pour lancer plusieurs instances d'apprentissage de modèles avec différentes configurations d'hyperparamètres afin de trouver le meilleur modèle pour une gamme de paramètres d'hyperparamètres. Consultez Personnalisation des configurations d'hyperparamètres de modèle dans Neptune ML.

Modèles d'intégration de graphe de connaissances dans Neptune ML

Les graphes de connaissances (KGs) sont des graphes qui encodent des informations sur les différentes entités (nœuds) et leurs relations (arêtes). Dans Neptune ML, les modèles d'intégration de graphe de connaissances sont appliqués par défaut pour effectuer une prédiction de lien lorsque le graphe ne contient pas de propriétés de nœud, mais uniquement des relations avec d'autres nœuds. Des modèles R-GCN avec intégrations pouvant être apprises peuvent également être utilisés pour ces graphes en spécifiant le type de modèle `"rgcn"`, mais les modèles d'intégration de graphe de connaissances sont plus simples et sont conçus pour être efficaces pour l'apprentissage de représentations pour des graphes de connaissances à grande échelle.

Les modèles d'intégration de graphe de connaissances sont utilisés dans une tâche de prédiction de lien pour prédire les nœuds ou les relations qui complètent un triple `(h, r, t)` où **`h`** est le nœud source, **`r`** le type de relation et **`t`** le nœud de destination.

Les modèles d'intégration de graphe de connaissances implémentés dans Neptune ML sont `distmult`, `transE` et `rotatE`. Pour en savoir plus sur les modèles d'intégration de graphe de connaissances, consultez DGL-KE.

Entraînement de modèles personnalisés dans Neptune ML

Neptune ML vous permet de définir et d'implémenter vos propres modèles personnalisés, pour des scénarios particuliers. Consultez Modèles personnalisés dans Neptune ML pour en savoir plus sur la manière d'implémenter un modèle personnalisé et sur la manière d'utiliser l'infrastructure Neptune ML pour l'entraîner.

Personnalisation des configurations d'hyperparamètres de modèle dans Neptune ML

Lorsque vous démarrez une tâche d'entraînement de modèle Neptune ML, Neptune ML utilise automatiquement les informations déduites de la tâche de traitement de données précédente. Il utilise les informations pour générer des plages de configuration d'hyperparamètres qui sont utilisées pour créer une tâche de réglage d'hyperparamètres basée sur l'SageMaker IA afin d'entraîner plusieurs modèles pour votre tâche. Ainsi, il n'est pas nécessaire de spécifier une longue liste de valeurs d'hyperparamètres à utiliser pour entraîner les modèles. Au lieu de cela, les valeurs par défaut et les plages d'hyperparamètres de modèle sont sélectionnées en fonction du type de tâche, du type de graphe et des paramètres de la tâche de réglage.

Toutefois, vous pouvez également remplacer la configuration des hyperparamètres par défaut et fournir des hyperparamètres personnalisés en modifiant un fichier de configuration JSON généré par la tâche de traitement des données.

À l'aide de l'API modelTraining de Neptune ML, vous pouvez contrôler plusieurs paramètres de tâche de réglage d'hyperparamètres de haut niveau, tels que `maxHPONumberOfTrainingJobs`, `maxHPOParallelTrainingJobs` et `trainingInstanceType`. Pour un contrôle plus précis des hyperparamètres du modèle, vous pouvez personnaliser le fichier `model-HPO-configuration.json` généré par la tâche de traitement de données. Le fichier est enregistré à l'emplacement Amazon S3 que vous avez spécifié pour la sortie de la tâche de traitement.

Vous pouvez télécharger le fichier, le modifier pour remplacer les configurations d'hyperparamètres par défaut, puis le charger à nouveau dans le même emplacement Amazon S3. Ne modifiez pas le nom du fichier et veillez à suivre ces instructions lorsque vous le modifiez.

Pour télécharger le fichier à partir d'Amazon S3 :

```
aws s3 cp \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json \
  ./
```

Lorsque vous avez terminé la modification, rechargez le fichier là où il se trouvait :

```
aws s3 cp \
  model-HPO-configuration.json \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json
```

Structure du fichier **model-HPO-configuration.json**

Le fichier `model-HPO-configuration.json` spécifie le modèle à entraîner, l'élément `task_type` de machine learning et les hyperparamètres qui doivent être modulés ou fixés pour les diverses exécutions d'entraînement de modèle.

Les hyperparamètres sont classés comme appartenant à différents niveaux qui indiquent la priorité accordée aux hyperparamètres lorsque la tâche de réglage des hyperparamètres est invoquée :

- Les hyperparamètres de niveau 1 ont la priorité la plus élevée. Si vous définissez `maxHPONumberOfTrainingJobs` sur une valeur inférieure à 10, seuls les hyperparamètres de niveau 1 sont réglés et les autres adoptent leurs valeurs par défaut.
- Les hyperparamètres de niveau 2 ont une priorité inférieure. Ainsi, si vous avez plus de 10 mais moins de 50 tâches d'entraînement au total pour une tâche de réglage, les hyperparamètres des niveaux 1 et 2 sont réglés.
- Les hyperparamètres de niveau 3 sont réglés avec les niveaux 1 et 2 uniquement si vous avez un total de plus de 50 tâches d'entraînement.
- Enfin, les hyperparamètres fixés ne sont pas du tout réglés et adoptent toujours leurs valeurs par défaut.

Exemple de fichier **model-HPO-configuration.json**

Voici un exemple de fichier `model-HPO-configuration.json` :

```
{
  "models": [
    {
      "model": "rgcn",
      "task_type": "node_class",
      "eval_metric": {
        "metric": "acc"
      },
      "eval_frequency": {
          "type":  "evaluate_every_epoch",
          "value":  1
      },
      "1-tier-param": [
        {
            "param": "num-hidden",
            "range": [16, 128],
```

```
            "type": "int",
            "inc_strategy": "power2"
        },
        {
          "param": "num-epochs",
          "range": [3,30],
          "inc_strategy": "linear",
          "inc_val": 1,
          "type": "int",
          "node_strategy": "perM"
        },
        {
          "param": "lr",
          "range": [0.001,0.01],
          "type": "float",
          "inc_strategy": "log"
        }
      ],
      "2-tier-param": [
        {
          "param": "dropout",
          "range": [0.0,0.5],
          "inc_strategy": "linear",
          "type": "float",
          "default": 0.3
        },
        {
          "param": "layer-norm",
          "type": "bool",
          "default": true
        }
      ],
      "3-tier-param": [
        {
          "param": "batch-size",
          "range": [128, 4096],
          "inc_strategy": "power2",
          "type": "int",
          "default": 1024
        },
        {
          "param": "fanout",
          "type": "int",
          "options": [[10, 30],[15, 30], [15, 30]],
```

```
          "default": [10, 15, 15]
        },
        {
          "param": "num-layer",
          "range": [1, 3],
          "inc_strategy": "linear",
          "inc_val": 1,
          "type": "int",
          "default": 2
        },
        {
          "param": "num-bases",
          "range": [0, 8],
          "inc_strategy": "linear",
          "inc_val": 2,
          "type": "int",
          "default": 0
        }
      ],
      "fixed-param": [
        {
          "param": "concat-node-embed",
          "type": "bool",
          "default": true
        },
        {
          "param": "use-self-loop",
          "type": "bool",
          "default": true
        },
        {
          "param": "low-mem",
          "type": "bool",
          "default": true
        },
        {
          "param": "l2norm",
          "type": "float",
          "default": 0
        }
      ]
    }
  ]
```

```
}
```

Éléments d'un fichier **`model-HPO-configuration.json`**

Le fichier contient un objet JSON avec un seul tableau de niveau supérieur nommé `models` contenant un seul objet de configuration de modèle. Lorsque vous personnalisez le fichier, assurez-vous que le tableau `models` ne contient qu'un seul objet de configuration de modèle. Si votre fichier contient plusieurs objets de configuration de modèle, la tâche de réglage échouera avec un avertissement.

L'objet de configuration de modèle contient les éléments de haut niveau suivants :

- **`model`** : (string) type de modèle à entraîner (ne pas modifier). Les valeurs valides sont :
  - `"rgcn"` : il s'agit de la valeur par défaut pour les tâches de classification et de régression de nœud, ainsi que pour les tâches de prédiction de lien hétérogènes.
  - `"transe"` : il s'agit de la valeur par défaut pour les tâches de prédiction de lien KGE.
  - `"distmult"` : il s'agit d'un type de modèle alternatif pour les tâches de prédiction de lien KGE.
  - `"rotate"` : il s'agit d'un type de modèle alternatif pour les tâches de prédiction de lien KGE.

  En règle générale, ne modifiez pas directement la valeur `model`, car les différents types de modèles ont souvent des hyperparamètres applicables sensiblement différents, ce qui peut entraîner une erreur d'analyse après le début de la tâche d'entraînement.

  Pour modifier le type de modèle, utilisez le paramètre `modelName` de l'API modelTraining au lieu de le modifier dans le fichier `model-HPO-configuration.json`.

  Une manière de modifier le type de modèle et d'apporter des modifications fines aux hyperparamètres consiste à copier le modèle de configuration de modèle par défaut pour le modèle que vous souhaitez utiliser et à le coller dans le fichier `model-HPO-configuration.json`. Il existe un dossier nommé `hpo-configuration-templates` au même emplacement Amazon S3 que le fichier `model-HPO-configuration.json` si le type de tâche déduit prend en charge plusieurs modèles. Ce dossier contient toutes les configurations d'hyperparamètres par défaut pour les autres modèles applicables à la tâche.

  Par exemple, si vous souhaitez modifier les configurations de modèle et d'hyperparamètres d'une tâche de prédiction de lien KGE en passant du modèle `transe` par défaut à un modèle `distmult`, il vous suffit de coller le contenu du fichier `hpo-configuration-templates/distmult.json`

dans le fichier `model-HPO-configuration.json`, puis d'apporter les modifications nécessaires aux hyperparamètres.

> **Note**
>
> Si vous définissez le paramètre `modelName` dans l'API `modelTraining` et modifiez également la spécification des hyperparamètres et de `model` dans le fichier `model-HPO-configuration.json`, et que ceux-ci sont différents, la valeur `model` du fichier `model-HPO-configuration.json` est prioritaire et la valeur `modelName` est ignorée.

- **`task_type`** : (string) type de tâche de machine learning déduit par la tâche de traitement des données ou transmis directement à cette tâche (ne pas modifier). Les valeurs valides sont :
  - `"node_class"`
  - `"node_regression"`
  - `"link_prediction"`

  La tâche de traitement des données déduit le type de tâche en examinant les propriétés du jeu de données exporté et du fichier généré de configuration de tâche d'entraînement.

  Cette valeur ne doit pas être modifiée. Si vous souhaitez entraîner une autre tâche, vous devez exécuter une nouvelle tâche de traitement de données. Si la valeur `task_type` n'est pas celle que vous attendiez, vous devez vérifier les entrées de votre tâche de traitement de données pour vous assurer qu'elles sont correctes. Cela inclut les paramètres de l'API `modelTraining`, ainsi que ceux du fichier de configuration de tâche d'entraînement généré par le processus d'exportation de données.
- **`eval_metric`** : (string) la métrique d'évaluation doit être utilisée pour évaluer les performances de modèle et pour sélectionner le modèle le plus performant lors des exécutions HPO. Les valeurs valides sont :
  - `"acc"` : précision de classification standard. Il s'agit de la valeur par défaut pour les tâches de classification à étiquette unique, sauf si des étiquettes déséquilibrées sont détectées lors du traitement des données, auquel cas la valeur par défaut est `"F1"`.
  - `"acc_topk"` : nombre de fois où l'étiquette correcte figure parmi les meilleures prédictions **k**. Vous pouvez également définir la valeur **k** en transmettant `topk` comme clé supplémentaire.
  - `"F1"` : le score F1.
  - `"mse"` : métrique d'erreur quadratique moyenne pour les tâches de régression.
  - `"mrr"` : métrique de rang réciproque moyen.

- `"precision"` : la précision du modèle, calculée comme le rapport entre les vrais positifs et les positifs prédits : `= true-positives / (true-positives + false-positives)`.
- `"recall"` : le rappel du modèle, calculé comme le rapport entre les vrais positifs et les positifs réels : `= true-positives / (true-positives + false-negatives)`.
- `"roc_auc"` : l'aire située sous la courbe ROC. Il s'agit de la valeur par défaut pour la classification à plusieurs étiquettes.

Par exemple, pour remplacer la métrique par `F1`, modifiez la valeur `eval_metric` comme suit :

```
"  eval_metric": {
    "metric": "F1",
  },
```

Ou, pour remplacer la métrique par un score de précision `topk`, vous devez modifier `eval_metric` comme suit :

```
  "eval_metric": {
    "metric": "acc_topk",
    "topk": 2
  },
```

- **`eval_frequency`** : (objet) spécifie à quelle fréquence il convient de vérifier les performances du modèle sur le jeu de validation pendant l'entraînement. Sur la base des performances de validation, il est alors possible d'initier un arrêt anticipé et d'enregistrer le meilleur modèle.

  L'objet `eval_frequency` contient deux éléments, à savoir `"type"` et `"value"`. Par exemple :

```
  "eval_frequency": {
    "type":  "evaluate_every_pct",
    "value":  0.1
  },
```

  Les valeurs `type` valides sont :

  - **`evaluate_every_pct`** : spécifie le pourcentage d'entraînement à suivre pour chaque évaluation.

    Pour `evaluate_every_pct`, le champ `"value"` contient un nombre à virgule flottante compris entre zéro et un, qui exprime ce pourcentage.

- **`evaluate_every_batch`** : spécifie le nombre de lots d'entraînement à traiter pour chaque évaluation.

  Pour `evaluate_every_batch`, le champ `"value"` contient un entier qui exprime ce nombre de lots.

- **`evaluate_every_epoch`** : spécifie le nombre d'époques par évaluation, une nouvelle époque commençant à minuit.

  Pour `evaluate_every_epoch`, le champ `"value"` contient un entier qui exprime ce nombre d'époques.

Le paramètre par défaut pour `eval_frequency` est :

```
"eval_frequency": {
  "type":  "evaluate_every_epoch",
  "value":  1
},
```

- **`1-tier-param`** : (obligatoire) tableau d'hyperparamètres de niveau 1.

  Si vous ne souhaitez régler aucun hyperparamètre, vous pouvez définir cet élément sur un tableau vide. Cela n'affecte pas le nombre total de tâches de formation lancées par la tâche de réglage des hyperparamètres de l' SageMaker IA. Cela signifie simplement que toutes les tâches d'entraînement, s'il y en a plus de 1 mais moins de 10, seront exécutées avec le même ensemble d'hyperparamètres.

  D'un autre côté, si vous souhaitez traiter tous vos hyperparamètres réglables avec une importance égale, vous pouvez placer tous les hyperparamètres dans ce tableau.

- **`2-tier-param`** : (obligatoire) tableau d'hyperparamètres de niveau 2.

  Ces paramètres sont réglés uniquement si `maxHPONumberOfTrainingJobs` a une valeur supérieure à 10. Dans le cas contraire, ils sont fixés aux valeurs par défaut.

  Si vous avez un budget d'entraînement de 10 tâches d'entraînement au maximum ou ne souhaitez pas d'hyperparamètres de niveau 2 pour une raison quelconque, mais que vous souhaitez régler tous les hyperparamètres réglables, vous pouvez définir cet élément sur un tableau vide.

- **`3-tier-param`** : (obligatoire) tableau d'hyperparamètres de niveau 3.

Ces paramètres sont réglés uniquement si `maxHPONumberOfTrainingJobs` a une valeur supérieure à 50. Dans le cas contraire, ils sont fixés aux valeurs par défaut.

Si vous ne souhaitez pas d'hyperparamètres de niveau 3, vous pouvez définir cet élément sur un tableau vide.

- **`fixed-param`** : (obligatoire) un tableau d'hyperparamètres fixes qui ne prennent que leurs valeurs par défaut et ne varient pas dans les différentes tâches d'entraînement.

  Si vous souhaitez moduler tous les hyperparamètres, vous pouvez définir cet élément sur un tableau vide et définir une valeur de `maxHPONumberOfTrainingJobs` suffisamment grande pour faire varier tous les niveaux ou définir tous les hyperparamètres au niveau 1.

L'objet JSON qui représente chaque hyperparamètre dans `1-tier-param`, `2-tier-param`, `3-tier-param` et `fixed-param` contient les éléments suivants :

- **`param`** : (string) nom de l'hyperparamètre (ne pas modifier).

  Consultez la liste des noms d'hyperparamètre valides dans Neptune ML.

- **`type`** : (string) type d'hyperparamètre (ne pas modifier).

  Les types valides sont : `bool`, `int` et `float`.

- **`default`** : (string) valeur par défaut de l'hyperparamètre.

  Vous pouvez définir une nouvelle valeur par défaut.

Les hyperparamètres réglables peuvent également contenir les éléments suivants :

- **`range`** : (array) plage d'un hyperparamètre réglable continu.

  Il doit s'agir d'un tableau à deux valeurs, à savoir le minimum et le maximum de la plage (`[min, max]`).

- **`options`** : (array) options pour un hyperparamètre réglable catégoriel.

  Ce tableau doit contenir toutes les options à prendre en compte :

  ```
  "options" : [value1, value2, ... valuen]
  ```

- **`inc_strategy`** : (string) type de modification incrémentielle pour les plages d'hyperparamètres réglables continus (ne pas modifier).

  Les valeurs valides sont `log`, `linear` et `power2`. Cela s'applique uniquement lorsque la clé de plage est définie.

  Si vous modifiez cet élément, vous risquez de ne pas utiliser toute la plage de votre hyperparamètre pour le réglage.
- **`inc_val`** : (float) différence entre les incréments successifs pour les hyperparamètres réglables continus (ne pas modifier).

  Cela s'applique uniquement lorsque la clé de plage est définie.

  Si vous modifiez cet élément, vous risquez de ne pas utiliser toute la plage de votre hyperparamètre pour le réglage.
- **`node_strategy`** : (string) indique que la plage effective de cet hyperparamètre doit changer en fonction du nombre de nœuds du graphe (ne pas modifier).

  Les valeurs valides sont `"perM"` (par million), `"per10M"` (pour 10 millions) et `"per100M"` (pour 100 millions).

  Plutôt que de modifier cette valeur, modifiez `range`.
- **`edge_strategy`** : (string) indique que la plage effective de cet hyperparamètre doit changer en fonction du nombre d'arêtes dans le graphe (ne pas modifier).

  Les valeurs valides sont `"perM"` (par million), `"per10M"` (pour 10 millions) et `"per100M"` (pour 100 millions).

  Plutôt que de modifier cette valeur, modifiez `range`.

Liste de tous les hyperparamètres dans Neptune ML

La liste suivante contient tous les hyperparamètres qui peuvent être définis n'importe où dans Neptune ML, pour tout type de modèle et toute tâche. Comme ils ne sont pas tous applicables à tout type de modèle, il est important de définir dans le fichier `model-HPO-configuration.json` seulement les hyperparamètres qui apparaissent dans le modèle relatif au modèle que vous utilisez.

- **`batch-size`** : taille du lot de nœuds cibles utilisé en un seul transfert. Type : `int`.

La définition d'une valeur beaucoup plus élevée peut entraîner des problèmes de mémoire lors de l'entraînement sur des instances GPU.

- **`concat-node-embed`** : indique s'il faut obtenir la représentation initiale d'un nœud en concaténant ses fonctionnalités traitées avec des intégrations initiales de nœuds pouvant être apprises afin d'augmenter l'expressivité du modèle. Type : `bool`.
- **`dropout`** : probabilité d'abandon appliquée aux couches d'abandon. Type : `float`.
- **`edge-num-hidden`** : taille de la couche masquée ou nombre d'unités pour le module de fonctionnalités d'arête. Utilisé uniquement quand `use-edge-features` a pour valeur `True`. Type : float.
- **`enable-early-stop`** : permet d'utiliser ou non la fonctionnalité d'arrêt anticipé. Type : `bool`. Par défaut : `true`.

  Utilisez ce paramètre booléen pour désactiver la fonctionnalité d'arrêt anticipé.
- **`fanout`** : nombre de voisins à échantillonner pour un nœud cible lors de l'échantillonnage des voisins. Type : `int`.

  Cette valeur est étroitement liée à `num-layers` et devrait toujours figurer au même niveau d'hyperparamètres. Cela est dû au fait que vous pouvez spécifier une diffusion en éventail pour chaque couche GNN potentielle.

  Étant donné que cet hyperparamètre peut entraîner des variations importantes des performances de modèle, il devrait être fixé ou défini en tant qu'hyperparamètre de niveau 2 ou 3. Lui affecter une valeur élevée peut entraîner des problèmes de mémoire lors de l'entraînement sur des instances GPU.
- **`gamma`** : valeur de marge dans la fonction de score. Type : `float`.

  Elle s'applique uniquement aux modèles de prédiction de lien `KGE`.
- **`l2norm`** : valeur de dégradation de poids utilisée dans l'optimiseur, qui impose une pénalité de normalisation L2 aux poids. Type : `bool`.
- **`layer-norm`** : indique s'il convient d'utiliser la normalisation des couches pour les modèles `rgcn`. Type : `bool`.
- **`low-mem`** : indique s'il convient d'utiliser une implémentation à faible mémoire de la fonction de transmission des messages relationnels au détriment de la vitesse. Type : `bool`.
- **`lr`** : taux d'apprentissage. Type : `float`.

Il doit être défini en tant qu'hyperparamètre de niveau 1.

- **`neg-share`** : dans le cadre de la prédiction de lien, indique si les arêtes échantillonnées positives peuvent partager des échantillons d'arête négatifs. Type : `bool`.
- **`num-bases`** : nombre de bases pour la décomposition des bases dans un modèle `rgcn`. L'utilisation d'une valeur `num-bases` inférieure au nombre de types d'arête dans le graphe agit comme un régularisateur pour le modèle `rgcn`. Type : `int`.
- **`num-epochs`** : nombre d'époques d'entraînement à exécuter. Type : `int`.

  Une époque est un parcours d'entraînement complet à travers le graphe.
- **`num-hidden`** : taille de la couche masquée ou nombre d'unités. Type : `int`.

  Cela définit également la taille d'intégration initiale pour les nœuds sans fonctionnalités.

  La définir sur une valeur beaucoup plus élevée sans la réduire `batch-size` peut entraîner des out-of-memory problèmes d'entraînement sur une instance GPU.
- **`num-layer`** : nombre de couches GNN dans le modèle. Type : `int`.

  Cette valeur est étroitement liée au paramètre de diffusion en éventail et devrait être définie après la diffusion en éventail au même niveau d'hyperparamètres.

  Étant donné qu'elle peut entraîner des variations importantes des performances de modèle, elle devrait être fixée ou définie en tant qu'hyperparamètre de niveau 2 ou 3.
- **`num-negs`** : dans le cadre de la prédiction de lien, nombre d'échantillons négatifs par échantillon positif. Type : `int`.
- **`per-feat-name-embed`** : indique s'il faut intégrer chaque fonctionnalité en la transformant indépendamment avant de combiner les fonctionnalités. Type : `bool`.

  Lorsque cette valeur est définie sur `true`, chaque fonctionnalité par nœud est transformée indépendamment en une taille de dimension fixe avant que toutes les fonctionnalités transformées pour le nœud soient concaténées puis transformées dans la dimension `num_hidden`.

  Lorsque la valeur `false` est définie, les fonctionnalités sont concaténées sans aucune transformation spécifique aux fonctionnalités.
- **`regularization-coef`** : dans le cadre de la prédiction de lien, coefficient de perte de régularisation. Type : `float`.

- **`rel-part`** : indique s'il convient d'utiliser une partition de relation pour la prédiction de lien `KGE`. Type : `bool`.
- **`sparse-lr`** : taux d'apprentissage pour les intégrations de nœuds pouvant être appris. Type : `float`.

  Les intégrations initiales de nœuds pouvant être appris sont utilisées pour les nœuds sans fonctionnalités ou quand `concat-node-embed` est défini. Les paramètres de la couche d'intégration fragmentée de nœuds pouvant être appris sont entraînés à l'aide d'un optimiseur distinct qui peut avoir un taux d'apprentissage distinct.
- **`use-class-weight`** : indique s'il convient d'appliquer des pondérations de classe pour les tâches de classification déséquilibrée. Si la valeur `true` est définie, les nombres d'étiquettes sont utilisés pour définir un poids pour chaque étiquette de classe. Type : `bool`.
- **`use-edge-features`** : indique s'il convient d'utiliser les fonctionnalités d'arête lors de la transmission de messages. Si la valeur `true` est définie, un module de fonctionnalités d'arête personnalisé est ajouté à la couche RGCN pour les types d'arête comportant des fonctionnalités. Type : `bool`.
- **`use-self-loop`** : indique s'il convient d'inclure les boucles automatiques dans l'entraînement d'un modèle `rgcn`. Type : `bool`.
- **`window-for-early-stop`** : contrôle le nombre des derniers scores de validation à la moyenne pour décider ou non d'un arrêt anticipé. La valeur par défaut est 3. type=int. Consultez également Arrêt anticipé du processus d'entraînement de modèle dans Neptune ML. Type : `int`. Par défaut : `3`.

  Consultez .

Personnalisation des hyperparamètres dans Neptune ML

Lorsque vous modifiez le fichier `model-HPO-configuration.json`, les types de modifications les plus courants sont les suivants :

- Modifiez les valeurs and/or maximales minimales des `range` hyperparamètres.
- Affectez une valeur fixe à un hyperparamètre en le déplaçant vers la section `fixed-param` et en définissant sa valeur par défaut sur la valeur fixe que vous souhaitez qu'il prenne.
- Modifiez la priorité d'un hyperparamètre en le plaçant dans un niveau spécifique, en modifiant sa plage et en vous assurant que sa valeur par défaut est définie de manière appropriée.

Bonnes pratiques d'entraînement de modèle

Il y a des choses que vous pouvez faire pour améliorer les performances des modèles Neptune ML.

Choisir la propriété de nœud appropriée

Toutes les propriétés de votre graphe ne sont pas significatives ou pertinentes pour vos tâches de machine learning. Toute propriété non pertinente doit être exclue lors de l'exportation des données.

Voici quelques bonnes pratiques :

- Faites appel à des experts du domaine pour vous aider à évaluer l'importance des fonctionnalités et la faisabilité de leur utilisation pour les prédictions.
- Supprimez les fonctionnalités que vous considérez redondantes ou non pertinentes afin de réduire le bruit dans les données et les corrélations superflues.
- Effectuez une itération au fur et à mesure que vous créez votre modèle. Ajustez les fonctionnalités, les combinaisons de fonctionnalités et les objectifs de réglage au fur et à mesure.

La rubrique Fonctionnalisation du Guide du développeur d'Amazon Machine Learning fournit des instructions supplémentaires relatives à la fonctionnalisation, qui sont pertinentes pour Neptune ML.

Gérer les points de données aberrants

Une aberration est un point de données très différent des autres données. Les données aberrantes peuvent gâcher ou induire en erreur le processus d'entraînement, ce qui peut conduire à un allongement du temps d'entraînement ou à des modèles moins précis. À moins qu'elles soient vraiment importantes, vous devez éliminer les aberrations avant d'exporter les données.

Supprimer les nœuds et les arêtes dupliqués

Les graphes stockés dans Neptune peuvent comporter des doublons de nœuds ou d'arêtes. Ces éléments redondants introduiront du bruit lors de l'entraînement de modèle ML. Éliminez les doublons de nœuds ou d'arêtes avant d'exporter les données.

Régler la structure du graphe

Lorsque le graphe est exporté, vous pouvez modifier la façon dont les fonctionnalités sont traitées et la façon dont le graphe est construit, afin d'améliorer les performances du modèle.

Voici quelques bonnes pratiques :

- Lorsqu'une propriété d'arête a la signification de catégories d'arête, il est parfois utile de la transformer en types d'arête.
- La politique de normalisation par défaut utilisée pour une propriété numérique est `min-max`, mais dans certains cas, d'autres politiques de normalisation fonctionnent mieux. Vous pouvez prétraiter la propriété et modifier la politique de normalisation comme expliqué dans Éléments d'un fichier `model-HPO-configuration.json`.
- Le processus d'exportation génère automatiquement des types de fonctionnalité en fonction des types de propriété. Par exemple, il traite les propriétés `String` comme des fonctionnalités catégorielles et les propriétés `Float` et `Int` comme des fonctionnalités numériques. Si nécessaire, vous pouvez modifier le type de fonctionnalité après l'exportation (voir Éléments d'un fichier `model-HPO-configuration.json`).

Régler les plages d'hyperparamètres et les valeurs par défaut

L'opération de traitement de données déduit des plages de configuration des hyperparamètres à partir du graphe. Si les plages d'hyperparamètres et les valeurs par défaut du modèle généré ne fonctionnent pas correctement pour vos données de graphe, vous pouvez modifier le fichier de configuration HPO afin de créer votre propre stratégie de réglage des hyperparamètres.

Voici quelques bonnes pratiques :

- Lorsque le graphe devient grand, la taille de la dimension masquée par défaut peut ne pas être suffisante pour contenir toutes les informations. Vous pouvez modifier l'hyperparamètre `num-hidden` pour contrôler la taille de la dimension masquée.
- Pour les modèles d'intégration de graphe de connaissances (KGE), vous souhaiterez peut-être modifier le modèle spécifique utilisé en fonction de votre structure de graphe et de votre budget.

  `TrainsE`les modèles ont du mal à gérer les relations one-to-many (1-N), many-to-one (N-1) et many-to-many (N-N). `DistMult`les modèles ont du mal à gérer les relations symétriques. `RotatE`est bon pour modéliser toutes sortes de relations, mais il est plus coûteux que `TrainsE` et `DistMult` pendant la formation.
- Dans certains cas, quand l'identification du nœud et les informations sur les fonctionnalités du nœud sont importantes, vous devez utiliser `` `concat-node-embed` `` pour indiquer au modèle Neptune ML d'obtenir la représentation initiale d'un nœud en concaténant ses fonctionnalités avec ses intégrations initiales.

- Lorsque vous obtenez des performances relativement bonnes sur certains hyperparamètres, vous pouvez ajuster l'espace de recherche des hyperparamètres en fonction de ces résultats.

Arrêt anticipé du processus d'entraînement de modèle dans Neptune ML

L'arrêt anticipé peut réduire de manière significative le temps d'entraînement de modèle et les coûts associés sans dégrader les performances de modèle. Il empêche également le surajustement du modèle aux données d'entraînement.

L'arrêt anticipé dépend de mesures régulières des performances du jeu de validation. Initialement, les performances s'améliorent au fur et à mesure de l'entraînement, mais lorsque le modèle commence à être surajusté, elles commencent à décliner à nouveau. La fonctionnalité d'arrêt anticipé identifie le point où le modèle commence à être surajusté et interrompt l'entraînement de modèle à ce stade.

Neptune ML surveille les appels des métriques de validation et compare la métrique de validation la plus récente à la moyenne des métriques de validation des **n** dernières évaluations, où **n** est un nombre défini à l'aide du paramètre `window-for-early-stop`. Dès que la métrique de validation est inférieure à cette moyenne, Neptune ML arrête l'entraînement du modèle et enregistre le meilleur modèle à ce stade.

Vous pouvez contrôler l'arrêt anticipé à l'aide des paramètres suivants :

- **`window-for-early-stop`** : la valeur de ce paramètre est un entier qui spécifie le nombre de scores de validation récents à moyenner pour décider d'un arrêt anticipé. La valeur par défaut est `3`.
- **`enable-early-stop`** : utilisez ce paramètre booléen pour désactiver la fonctionnalité d'arrêt anticipé. Par défaut, cette valeur est `true`.

Arrêt anticipé du processus HPO dans Neptune ML

La fonction d'arrêt anticipé de Neptune ML arrête également les tâches d'entraînement qui ne fonctionnent pas bien par rapport aux autres tâches de formation, à l'aide de la fonction de démarrage à chaud SageMaker AI HPO. Cela peut également réduire les coûts et améliorer la qualité de l'optimisation des hyperparamètres (HPO).

Consultez Exécution d'une tâche de réglage des hyperparamètres avec démarrage à chaud pour obtenir une description de son fonctionnement.

Le démarrage à chaud permet de transmettre les informations apprises à partir des tâches d'entraînement précédentes aux tâches d'entraînement suivantes et offre deux avantages distincts :

- Tout d'abord, les résultats des tâches d'entraînement précédentes sont utilisés pour sélectionner de bonnes combinaisons d'hyperparamètres à examiner dans la nouvelle tâche de réglage.
- Ensuite, il permet un arrêt anticipé pour accéder à un plus grand nombre d'exécutions de modèle, ce qui réduit le temps de réglage.

Cette fonctionnalité est activée automatiquement dans Neptune ML et vous permet de trouver un équilibre entre le temps d'entraînement de modèle et ses performances. Si vous êtes satisfait des performances du modèle actuel, vous pouvez utiliser ce modèle. Sinon, vous en exécutez d'autres HPOs qui démarrent à chaud avec les résultats des essais précédents afin de découvrir un meilleur modèle.

Bénéficier de services d'assistance professionnels

AWS propose des services de support professionnels pour vous aider à résoudre les problèmes liés à vos projets d'apprentissage automatique sur Neptune. Si vous êtes bloqué, contactez AWS Support.

Utilisation d'un modèle entraîné pour générer de nouveaux artefacts de modèle

À l'aide de la commande de transformation de modèle Neptune ML, vous pouvez calculer des artefacts de modèle tels que les intégrations de nœuds sur des données de graphe traitées à l'aide de paramètres de modèle préentraînés.

Transformation de modèle pour une inférence incrémentielle

Dans le flux de travail d'inférence de modèle incrémentielle, après avoir traité les données graphiques mises à jour que vous avez exportées depuis Neptune, vous pouvez démarrer une tâche de transformation de modèle à l'aide d'une commande comme celle-ci :

AWS CLI

```
aws neptunedata start-ml-model-transform-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique model-transform job ID)" \
  --data-processing-job-id "(the data-processing job-id of a completed job)" \
  --ml-model-training-job-id "(the ML model training job-id)" \
  --model-transform-output-s3-location "s3://(your S3 bucket)/neptune-model-
transform/"
```

Pour plus d'informations, consultez start-ml-model-transform-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_model_transform_job(
    id='(a unique model-transform job ID)',
    dataProcessingJobId='(the data-processing job-id of a completed job)',
```

```
    mlModelTrainingJobId='(the ML model training job-id)',
    modelTransformOutputS3Location='s3://(your S3 bucket)/neptune-model-transform/'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltransform \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "mlModelTrainingJobId": "(the ML model training job-id)",
        "modelTransformOutputS3Location" : "s3://(your S3 bucket)/neptune-model-
transform/"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/modeltransform \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "mlModelTrainingJobId": "(the ML model training job-id)",
        "modelTransformOutputS3Location" : "s3://(your S3 bucket)/neptune-model-
transform/"
      }'
```

Vous pouvez ensuite transmettre l'ID de cette tâche à l'appel d'API de création de points de terminaison pour créer un nouveau point de terminaison ou mettre à jour un point de terminaison existant avec les nouveaux artefacts de modèle générés par cette tâche. Cela permet au point de terminaison nouveau ou mis à jour de fournir des prédictions de modèle pour les données de graphe mises à jour.

Transformation de modèle pour n'importe quelle tâche d'entraînement

Vous pouvez également fournir un `trainingJobName` paramètre pour générer des artefacts de modèle pour toutes les tâches de formation à l' SageMaker IA lancées pendant l'entraînement du modèle Neptune ML. Étant donné qu'une tâche de formation sur le modèle Neptune ML peut potentiellement lancer de nombreuses tâches de formation à l' SageMaker IA, cela vous donne la flexibilité de créer un point de terminaison d'inférence basé sur l'une de ces tâches de formation à l' SageMaker IA.

Par exemple :

AWS CLI

```
aws neptunedata start-ml-model-transform-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique model-transform job ID)" \
  --training-job-name "(name of a completed SageMaker training job)" \
  --model-transform-output-s3-location "s3://(your S3 bucket)/neptune-model-
transform/"
```

Pour plus d'informations, consultez start-ml-model-transform-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_model_transform_job(
    id='(a unique model-transform job ID)',
```

```
    trainingJobName='(name of a completed SageMaker training job)',
    modelTransformOutputS3Location='s3://(your S3 bucket)/neptune-model-transform/'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltransform \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your S3 bucket)/neptune-model-
transform/"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/modeltransform \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your S3 bucket)/neptune-model-
transform/"
      }'
```

Si la tâche d'entraînement d'origine portait sur un modèle personnalisé fourni par l'utilisateur, vous devez inclure un objet `customModelTransformParameters` lorsque vous invoquez une

transformation de modèle. Consultez Modèles personnalisés dans Neptune ML pour en savoir plus sur la façon d'implémenter et d'utiliser un modèle personnalisé.

> **Note**
>
> La `modeltransform` commande exécute toujours la transformation du modèle sur la tâche de formation à l' SageMaker IA la mieux adaptée à cette formation.

Consultez Commande modeltransform pour plus d'informations sur les tâches de transformation de modèle.

Artefacts produits par l'entraînement de modèle dans Neptune ML

Après l'entraînement de modèle, Neptune ML utilise les paramètres de modèle les mieux entraînés pour générer les artefacts de modèle qui sont nécessaires pour le lancement du point de terminaison d'inférence et la fourniture de prédictions de modèle. Ces artefacts sont regroupés par la tâche de formation et stockés dans l'emplacement de sortie Amazon S3 de la meilleure tâche de formation en SageMaker IA.

Les sections suivantes décrivent ce qui est inclus dans les artefacts de modèle pour les différentes tâches et comment la commande de transformation de modèle utilise un modèle entraîné préexistant pour générer des artefacts même sur de nouvelles données de graphe.

Artefacts générés pour différentes tâches

Le contenu des artefacts de modèle générés par le processus d'entraînement dépend de la tâche de machine learning cible :

- Classification et régression de nœud : pour la prédiction d'une propriété d'un nœud, les artefacts incluent les paramètres de modèle, les intégrations de nœud provenant de l'encodeur GNN, les prédictions de modèle pour les nœuds figurant dans le graphe d'entraînement et certains fichiers de configuration du point de terminaison d'inférence. Dans les tâches de classification et de régression de nœud, les prédictions de modèle sont précalculées pour les nœuds présents pendant l'entraînement afin de réduire la latence des requêtes.
- Classification et régression d'arête : pour la prédiction d'une propriété d'une arête, les artefacts incluent également les paramètres de modèle et les intégrations de nœud. Les paramètres du décodeur de modèle sont particulièrement importants pour l'inférence, car nous calculons les prédictions de classification ou de régression d'arête en appliquant le décodeur de modèle aux intégrations des sommets source et de destination d'une arête.
- Prédiction de lien : pour la prédiction de lien, outre les artefacts générés pour la prédiction d'une propriété d'une arête, le graphe DGL est également inclus en tant qu'artefact, car la prédiction de lien nécessite le graphe d'entraînement pour effectuer des prédictions. L'objectif de la prédiction de lien est de prédire les sommets de destination susceptibles de se combiner avec un sommet source pour former une arête d'un type particulier dans le graphe. Pour ce faire, l'intégration de nœud du sommet source et une représentation apprise pour le type d'arête sont combinées avec les intégrations de nœuds de tous les sommets de destination possibles, afin de produire un score de probabilité d'arête pour chacun des sommets de destination. Les scores sont ensuite triés pour classer les sommets de destination potentiels et renvoyer les meilleurs candidats.

Pour chacun des types de tâche, les poids du modèle de réseau de neurones en graphes issu de la bibliothèque DGL sont enregistrés dans l'artefact du modèle. Cela permet à Neptune ML de calculer de nouvelles sorties du modèle à mesure que le graphe change (inférence inductive), en plus d'utiliser des prédictions et des intégrations précalculées (inférence transductive) pour réduire la latence.

Génération de nouveaux artefacts de modèle

Les artefacts de modèle générés après l'entraînement de modèle dans Neptune ML sont directement liés au processus d'entraînement. Cela signifie que les intégrations et les prédictions précalculées n'existent que pour les entités qui figuraient dans le graphe d'entraînement d'origine. Bien que le mode d'inférence inductive pour les points de terminaison Neptune ML puisse calculer des prédictions pour de nouvelles entités en temps réel, vous souhaiterez peut-être générer des prédictions par lots sur de nouvelles entités sans interroger un point de terminaison.

Afin d'obtenir des prédictions de modèle par lots pour de nouvelles entités ajoutées au graphe, les nouveaux artefacts de modèle doivent être recalculés pour les nouvelles données de graphe. Cela se fait à l'aide de la commande `modeltransform`. Vous utilisez la commande `modeltransform` lorsque vous souhaitez uniquement des prédictions par lots sans configurer de point de terminaison, ou lorsque vous souhaitez générer toutes les prédictions afin de pouvoir les réécrire dans le graphe.

Étant donné que l'entraînement de modèle effectue implicitement une transformation de modèle à la fin du processus d'entraînement, les artefacts de modèle sont toujours recalculés sur les données de graphe d'entraînement par une tâche d'entraînement. Toutefois, la commande `modeltransform` peut également calculer des artefacts de modèle sur des données de graphe qui n'ont pas été utilisées pour l'entraînement d'un modèle. Pour ce faire, les nouvelles données de graphe doivent être traitées à l'aide des mêmes encodages de fonctionnalité que les données de graphe d'origine et doivent adhérer au même schéma de graphe.

Vous pouvez effectuer cela en commençant par créer une nouvelle tâche de traitement de données qui constitue un clone de la tâche de traitement de données exécutée sur les données de graphe d'entraînement d'origine, puis en l'exécutant sur les nouvelles données de graphe (voir Traitement de données de graphe mises à jour pour Neptune ML). Ensuite, appelez la commande `modeltransform` avec le nouvel identifiant `dataProcessingJobId` et l'ancien identifiant `modelTrainingJobId` pour recalculer les artefacts de modèle sur les données de graphe mises à jour.

Pour la prédiction d'une propriété d'un nœud, les intégrations et les prédictions de nœud sont recalculées sur les nouvelles données de graphe, même pour les nœuds présents dans le graphe d'entraînement d'origine.

Pour la prédiction d'une propriété d'arête et la prédiction de lien, les intégrations de nœud sont également recalculées et remplacent de la même manière toutes les intégrations de nœud existantes. Pour recalculer les intégrations de nœud, Neptune ML applique l'encodeur GNN appris à partir du modèle entraîné précédent aux nœuds des nouvelles données de graphe avec leurs nouvelles fonctionnalités.

Pour les nœuds dépourvus de fonctionnalités, les représentations initiales apprises à partir de l'entraînement de modèle d'origine sont réutilisées. Pour les nouveaux nœuds dépourvus de fonctionnalités et qui n'étaient pas présents dans le graphe d'entraînement d'origine, Neptune ML initialise leur représentation comme la moyenne des représentations de nœud initiales apprises de ce type de nœud présentes dans le graphe d'entraînement d'origine. Cela peut entraîner une baisse des performances dans les prédictions de modèle si de nombreux nouveaux nœuds n'ont pas de fonctionnalités, car ils seront tous initialisés selon l'intégration initiale moyenne pour ce type de nœud.

Si votre modèle est entraîné avec `concat-node-embed` défini sur true, les représentations de nœud initiales sont créées en concaténant les fonctionnalités de nœud avec la représentation initiale pouvant être apprise. Ainsi, pour le graphe mis à jour, la représentation de nœud initiale des nouveaux nœuds utilise également les intégrations de nœud initiales moyennes, concaténées avec les nouvelles fonctionnalités de nœud.

En outre, les suppressions de nœuds ne sont actuellement pas prises en charge. Si des nœuds ont été supprimés dans le graphe mis à jour, vous devez réentraîner le modèle sur les données de graphe mises à jour.

Le recalcul des artefacts de modèle réutilise les paramètres de modèle appris sur un nouveau graphe et doit être effectué uniquement lorsque le nouveau graphe est très similaire à l'ancien. Si votre nouveau graphe n'est pas suffisamment similaire, vous devez réentraîner le modèle pour obtenir des performances de modèle similaires sur les nouvelles données de graphe. Ce qui constitue une similitude suffisante dépend de la structure de vos données de graphe, mais en règle générale, vous devez réentraîner votre modèle si vos nouvelles données diffèrent de plus de 10 à 20 % des données de graphe d'entraînement d'origine.

Pour les graphes où tous les nœuds ont des fonctionnalités, le seuil supérieur (20 % de différence) s'applique, mais pour les graphes où de nombreux nœuds n'ont pas de fonctionnalités et où les

nouveaux nœuds ajoutés au graphe n'ont pas de propriétés, le seuil inférieur (10 % de différence) peut même s'avérer trop élevé.

Consultez Commande modeltransform pour plus d'informations sur les tâches de transformation de modèle.

Modèles personnalisés dans Neptune ML

> **Note**
>
> La prise en charge des modèles personnalisés de Neptune ML repose sur une ancienne version de Python 3. Pour créer et exécuter des modèles GNN personnalisés avec des up-to-date dépendances, utilisez GraphStorm on SageMaker.
> L'inférence inductive en temps réel n'est actuellement pas prise en charge pour les modèles personnalisés.

Neptune ML vous permet de définir vos propres implémentations de modèle personnalisé à l'aide de Python. Vous pouvez entraîner et déployer des modèles personnalisés à l'aide de l'infrastructure Neptune ML d'une manière très similaire à celle employée pour les modèles intégrés, et vous pouvez les utiliser pour obtenir des prédictions par le biais de requêtes de graphe.

Vous pouvez commencer à implémenter votre propre modèle personnalisé en Python en suivant les exemples de la boîte à outils Neptune ML et en utilisant les composants de modèle fournis dans la boîte à outils Neptune ML. Les sections suivantes fournissent davantage de détails.

Table des matières

- Vue d'ensemble des modèles personnalisés dans Neptune ML
  - Quand utiliser un modèle personnalisé dans Neptune ML
  - Flux de travail pour le développement et l'utilisation d'un modèle personnalisé dans Neptune ML
- Développement de modèle personnalisé dans Neptune ML
  - Développement d'un script d'entraînement de modèle personnalisé dans Neptune ML
  - Développement d'un script de transformation de modèle personnalisé dans Neptune ML
  - Fichier model-hpo-configuration.json personnalisé dans Neptune ML
  - Test local de l'implémentation de votre modèle personnalisé dans Neptune ML

Vue d'ensemble des modèles personnalisés dans Neptune ML

Quand utiliser un modèle personnalisé dans Neptune ML

Les modèles intégrés de Neptune ML traitent toutes les tâches standard prises en charge par Neptune ML. Toutefois, dans certains cas, vous pouvez souhaiter disposer d'un contrôle plus précis sur le modèle pour une tâche particulière ou avoir besoin de personnaliser le processus d'entraînement de modèle. Un modèle personnalisé est par exemple approprié dans les cas suivants :

- L'encodage des fonctionnalités pour les fonctionnalités de texte de très grands modèles de texte doit être exécuté sur le GPU.
- Vous souhaitez utiliser votre propre modèle de réseau de neurones en graphes (GNN) personnalisé, développé dans la bibliothèque Deep Graph Library (DGL).
- Vous souhaitez utiliser des modèles tabulaires ou des modèles d'ensemble pour la classification et la régression de nœud.

Flux de travail pour le développement et l'utilisation d'un modèle personnalisé dans Neptune ML

La prise en charge des modèles personnalisés dans Neptune ML est conçue pour s'intégrer parfaitement aux flux de travail Neptune ML existants. Elle fonctionne en exécutant du code personnalisé dans votre module source sur l'infrastructure de Neptune ML pour entraîner le modèle. Comme c'est le cas pour un mode intégré, Neptune ML lance automatiquement une tâche de HyperParameter réglage par SageMaker IA et sélectionne le meilleur modèle en fonction de la métrique d'évaluation. Il utilise ensuite l'implémentation fournie dans votre module source pour générer des artefacts de modèle à déployer.

L'exportation des données, la configuration de l'entraînement et le prétraitement des données sont les mêmes pour un modèle personnalisé que pour un modèle intégré.

Après le prétraitement des données, vous pouvez développer et tester de manière itérative et interactive l'implémentation de votre modèle personnalisé à l'aide de Python. Lorsque votre modèle est prêt pour la production, vous pouvez charger le module Python obtenu sur Amazon S3 comme suit :

```
aws s3 cp --recursive (source path to module) s3://(bucket name)/(destination path for
 your module)
```

Vous pouvez ensuite utiliser le flux de travail de données normal par défaut ou incrémentiel pour déployer le modèle en production, à quelques différences près.

Pour l'entraînement de modèle à l'aide d'un modèle personnalisé, vous devez fournir un objet JSON `customModelTrainingParameters` à l'API d'entraînement de modèle de Neptune ML afin de garantir que votre code personnalisé est utilisé. Les champs présents dans l'objet `customModelTrainingParameters` sont les suivants :

- **`sourceS3DirectoryPath`** : (obligatoire) chemin de l'emplacement Amazon S3 où se trouve le module Python implémentant votre modèle. Il doit pointer vers un emplacement Amazon S3 existant valide contenant, au minimum, un script d'entraînement, un script de transformation et un fichier `model-hpo-configuration.json`.
- **`trainingEntryPointScript`** : (facultatif) nom du point d'entrée dans votre module d'un script qui effectue l'entraînement de modèle et utilise les hyperparamètres comme arguments de ligne de commande, y compris les hyperparamètres fixes.

  Par défaut : `training.py`.
- **`transformEntryPointScript`** : (facultatif) nom du point d'entrée dans le module d'un script qui doit être exécuté une fois que le modèle le plus approprié issu de la recherche par hyperparamètres a été identifié, afin de calculer les artefacts de modèle nécessaires au déploiement du modèle. Il devrait pouvoir s'exécuter sans arguments de ligne de commande.

  Par défaut : `transform.py`.

Par exemple :

AWS CLI

```
aws neptunedata start-ml-model-training-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique model-training job ID)" \
  --data-processing-job-id "(the data-processing job-id of a completed job)" \
  --train-model-s3-location "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer" \
  --model-name "custom" \
  --custom-model-training-parameters '{
    "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
    "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
```

```
    "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
  }'
```

Pour plus d'informations, consultez start-ml-model-training-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_model_training_job(
    id='(a unique model-training job ID)',
    dataProcessingJobId='(the data-processing job-id of a completed job)',
    trainModelS3Location='s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer',
    modelName='custom',
    customModelTrainingParameters={
        'sourceS3DirectoryPath': 's3://(your Amazon S3 bucket)/(path to your Python
 module)',
        'trainingEntryPointScript': '(your training script entry-point name in the
 Python module)',
        'transformEntryPointScript': '(your transform script entry-point name in the
 Python module)'
    }
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltraining \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
```

```
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer",
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your
 Python module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in
 the Python module)"
        }
     }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/modeltraining \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer",
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your
 Python module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in
 the Python module)"
        }
     }'
```

De même, pour activer une transformation de modèle personnalisée, vous devez fournir un objet JSON `customModelTransformParameters` à l'API de transformation de modèle de Neptune ML, avec des valeurs de champs compatibles avec les paramètres de modèle enregistrés à partir de la tâche d'entraînement. L'objet `customModelTransformParameters` contient les champs suivants :

- **`sourceS3DirectoryPath`** : (obligatoire) chemin de l'emplacement Amazon S3 où se trouve le module Python implémentant votre modèle. Il doit pointer vers un emplacement Amazon S3 existant valide contenant, au minimum, un script d'entraînement, un script de transformation et un fichier `model-hpo-configuration.json`.
- **`transformEntryPointScript`** : (facultatif) nom du point d'entrée dans le module d'un script qui doit être exécuté une fois que le modèle le plus approprié issu de la recherche par hyperparamètres a été identifié, afin de calculer les artefacts de modèle nécessaires au déploiement du modèle. Il devrait pouvoir s'exécuter sans arguments de ligne de commande.

  Par défaut : `transform.py`.

Par exemple :

AWS CLI

```
aws neptunedata start-ml-model-transform-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique model-transform job ID)" \
  --training-job-name "(name of a completed SageMaker training job)" \
  --model-transform-output-s3-location "s3://(your Amazon S3 bucket)/neptune-model-
transform/" \
  --custom-model-transform-parameters '{
    "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
    "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
  }'
```

Pour plus d'informations, consultez start-ml-model-transform-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config
```

```
client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_model_transform_job(
    id='(a unique model-transform job ID)',
    trainingJobName='(name of a completed SageMaker training job)',
    modelTransformOutputS3Location='s3://(your Amazon S3 bucket)/neptune-model-
transform/',
    customModelTransformParameters={
        'sourceS3DirectoryPath': 's3://(your Amazon S3 bucket)/(path to your Python
 module)',
        'transformEntryPointScript': '(your transform script entry-point name in the
 Python module)'
    }
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltransform \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-
model-transform/",
        "customModelTransformParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in
 the Python module)"
        }
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/modeltransform \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-
model-transform/",
        "customModelTransformParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in
 the Python module)"
        }
      }'
```

Développement de modèle personnalisé dans Neptune ML

Une bonne manière de démarrer le développement de modèle personnalisé consiste à suivre les exemples de la boîte à outils Neptune ML pour structurer et écrire votre module d'entraînement. La boîte à outils Neptune ML implémente également des composants de modèle ML de graphe modulaire dans l'élément modelzoo que vous pouvez empiler et utiliser pour créer votre modèle personnalisé.

En outre, la boîte à outils fournit des fonctions utilitaires qui vous aident à générer les artefacts nécessaires pendant l'entraînement de modèle et la transformation de modèle. Vous pouvez importer ce package Python dans votre implémentation personnalisée. Toutes les fonctions ou tous les modules fournis dans la boîte à outils sont également disponibles dans l'environnement d'entraînement Neptune ML.

Si votre module Python possède des dépendances externes supplémentaires, vous pouvez les inclure en créant un fichier `requirements.txt` dans le répertoire de votre module. Les packages répertoriés dans le fichier `requirements.txt` seront ensuite installés avant l'exécution de votre script d'entraînement.

Au minimum, le module Python qui implémente votre modèle personnalisé doit contenir les éléments suivants :

- Un point d'entrée de script d'entraînement
- Un point d'entrée de script de transformation
- Un fichier `model-hpo-configuration.json`

Développement d'un script d'entraînement de modèle personnalisé dans Neptune ML

Votre script d'entraînement de modèle personnalisé doit être un script Python exécutable, comme l'exemple `train.py` de la boîte à outils Neptune ML. Il doit accepter les noms et les valeurs des hyperparamètres en tant qu'arguments de ligne de commande. Pendant l'entraînement de modèle, les noms des hyperparamètres sont obtenus à partir du fichier `model-hpo-configuration.json`. Les valeurs des hyperparamètres se situent dans la plage des hyperparamètres valides si l'hyperparamètre est réglable, ou prennent la valeur d'hyperparamètre par défaut s'il n'est pas réglable.

Votre script d'entraînement est exécuté sur une instance d'entraînement SageMaker AI en utilisant une syntaxe comme celle-ci :

```
python3 (script entry point) --(1st parameter) (1st value) --(2nd parameter) (2nd
 value) (...)
```

Pour toutes les tâches, Neptune ML AutoTrainer envoie plusieurs paramètres requis à votre script d'entraînement en plus des hyperparamètres que vous spécifiez, et votre script doit être capable de gérer ces paramètres supplémentaires pour fonctionner correctement.

Ces paramètres obligatoires supplémentaires varient quelque peu en fonction de la tâche :

Pour la classification de nœud ou la régression de nœud

- **`task`** : type de tâche utilisé en interne par Neptune ML. Pour la classification de nœud, il s'agit de `node_class` et, pour la régression de nœud, il s'agit de `node_regression`.
- **`model`** : nom du modèle utilisé en interne par Neptune ML, à savoir `custom` dans ce cas.
- **`name`** : nom de la tâche utilisée en interne par Neptune ML, à savoir `node_class-custom` pour la classification de nœud dans ce cas, et `node_regression-custom` pour la régression de nœud.
- **`target_ntype`** : nom du type de nœud pour la classification ou la régression.
- **`property`** : nom de la propriété de nœud pour la classification ou la régression.

Pour la prédiction de lien

- **`task`** : type de tâche utilisé en interne par Neptune ML. Pour la prédiction de lien, il s'agit de `link_predict`.
- **`model`** : nom du modèle utilisé en interne par Neptune ML, à savoir `custom` dans ce cas.
- **`name`** : nom de la tâche utilisée en interne par Neptune ML, à savoir `link_predict-custom` dans ce cas.

Pour la classification d'arête ou la régression d'arête

- **`task`** : type de tâche utilisé en interne par Neptune ML. Pour la classification d'arête, il s'agit de `edge_class` et, pour la régression d'arête, il s'agit de `edge_regression`.
- **`model`** : nom du modèle utilisé en interne par Neptune ML, à savoir `custom` dans ce cas.
- **`name`** : nom de la tâche utilisée en interne par Neptune ML, à savoir `edge_class-custom` pour la classification d'arête dans ce cas, et `edge_regression-custom` pour la régression d'arête.
- **`target_etype`** : nom du type d'arête pour la classification ou la régression.

- **`property`** : nom de la propriété d'arête pour la classification ou la régression.

Votre script doit enregistrer les paramètres du modèle, ainsi que tous les autres artefacts qui seront nécessaires à la fin de l'entraînement.

Vous pouvez utiliser les fonctions utilitaires de la boîte à outils Neptune ML pour déterminer l'emplacement des données de graphe traitées, l'emplacement où les paramètres de modèle doivent être enregistrés et les périphériques GPU disponibles sur l'instance d'entraînement. Consultez l'exemple de script d'entraînement train.py pour des exemples d'utilisation de ces fonctions utilitaires.

Développement d'un script de transformation de modèle personnalisé dans Neptune ML

Un script de transformation est nécessaire pour tirer parti du flux de travail incrémentiel Neptune ML pour l'inférence de modèle sur des graphes évolutifs sans avoir à réentraîner le modèle. Même si tous les artefacts nécessaires au déploiement de modèle sont générés par le script d'entraînement, vous devez tout de même fournir un script de transformation si vous souhaitez générer des modèles mis à jour sans réentraîner le modèle.

> **Note**
>
> L'inférence inductive en temps réel n'est actuellement pas prise en charge pour les modèles personnalisés.

Votre script de transformation de modèle personnalisé doit être un script Python exécutable, comme l'exemple de script transform.py de la boîte à outils Neptune ML. Comme ce script est invoqué pendant l'entraînement de modèle sans arguments de ligne de commande, tous les arguments de ligne de commande acceptés par le script doivent avoir les valeurs par défaut.

Le script s'exécute sur une instance d'entraînement SageMaker AI avec une syntaxe comme celle-ci :

```
python3 (your transform script entry point)
```

Votre script de transformation aura besoin de diverses informations, telles que :

- L'emplacement des données du graphe traité.
- L'emplacement où les paramètres du modèle sont enregistrés et où les nouveaux artefacts de modèle doivent être enregistrés.

- Les appareils disponibles sur l'instance.
- Les hyperparamètres qui ont généré le meilleur modèle.

Ces entrées sont obtenues à l'aide des fonctions utilitaires Neptune ML que votre script peut appeler. Consultez l'exemple de script transform.py de la boîte à outils pour obtenir des exemples de la manière de procéder.

Le script doit enregistrer les intégrations de nœuds, les mappages d'ID de nœud et tous les autres artefacts nécessaires au déploiement du modèle pour chaque tâche. Consultez la documentation sur les artefacts de modèle pour plus d'informations sur les artefacts de modèle requis pour les différentes tâches Neptune ML.

Fichier **`model-hpo-configuration.json`** personnalisé dans Neptune ML

Le fichier `model-hpo-configuration.json` définit les hyperparamètres de votre modèle personnalisé. Il est au même format que le fichier `model-hpo-configuration.json` utilisé avec les modèles intégrés de Neptune ML et il a priorité sur la version générée automatiquement par Neptune ML et chargée à l'emplacement de vos données traitées.

Lorsque vous ajoutez un nouvel hyperparamètre à votre modèle, vous devez également ajouter une entrée pour l'hyperparamètre dans ce fichier afin que l'hyperparamètre soit transmis à votre script d'entraînement.

Vous devez fournir une plage pour un hyperparamètre si vous souhaitez qu'il soit réglable et le définir en tant que paramètre `tier-1`, `tier-2` ou `tier-3`. L'hyperparamètre sera réglé si le nombre total de tâches d'entraînement configurées permet de régler les hyperparamètres de son niveau. Pour un paramètre non réglable, vous devez fournir une valeur par défaut et ajouter l'hyperparamètre dans la section `fixed-param` du fichier. Consultez le fichier d'exemple `model-hpo-configuration.json` de la boîte à outils pour obtenir un exemple de la procédure à suivre.

Vous devez également fournir la définition de métrique que la tâche HyperParameter d'optimisation de l' SageMaker IA utilisera pour évaluer les modèles candidats entraînés. Pour ce faire, vous devez ajouter un objet JSON `eval_metric` dans le fichier `model-hpo-configuration.json` comme suit :

```
"eval_metric": {
  "tuning_objective": {
      "MetricName": "(metric_name)",
      "Type": "Maximize"
```

```
  },
  "metric_definitions": [
    {
      "Name": "(metric_name)",
      "Regex": "(metric regular expression)"
    }
  ]
},
```

Le `metric_definitions` tableau de l'`eval_metric`objet répertorie les objets de définition des métriques pour chaque métrique que vous souhaitez que l' SageMaker IA extrait de l'instance d'apprentissage. Chaque objet de définition de métrique possède une clé `Name` qui vous permet de donner un nom à la métrique (par exemple « précision », « f1 », etc.). La clé `Regex` vous permet de fournir une chaîne d'expression régulière correspondant à la manière dont cette métrique particulière est imprimée dans les journaux d'entraînement. Consultez la page SageMaker AI HyperParameter Tuning pour plus de détails sur la définition des métriques.

L'objet `tuning_objective` dans `eval_metric` vous permet ensuite de spécifier quelle métrique de `metric_definitions` doit être utilisée comme métrique d'évaluation servant de métrique d'objectif pour l'optimisation des hyperparamètres. La valeur de `MetricName` doit correspondre à la valeur d'un élément `Name` dans l'une des définitions de `metric_definitions`. La valeur de `Type` doit être « Maximiser » ou « Minimiser » selon que la métrique doit être interprétée comme greater-is-better (comme « précision ») ou less-is-better (comme « mean-squared-error »).

Des erreurs dans cette section du `model-hpo-configuration.json` fichier peuvent entraîner l'échec de la tâche d'API d'entraînement du modèle Neptune ML, car la tâche SageMaker AI HyperParameter Tuning ne sera pas en mesure de sélectionner le meilleur modèle.

Test local de l'implémentation de votre modèle personnalisé dans Neptune ML

Vous pouvez utiliser l'environnement Conda de la boîte à outils Neptune ML pour exécuter votre code localement afin de tester et de valider votre modèle. Si vous effectuez le développement sur une instance de bloc-notes Neptune, cet environnement Conda sera préinstallé sur l'instance de bloc-notes Neptune. Si vous effectuez le développement sur une autre instance, vous devez suivre les instructions de configuration locale (langue française non garantie) de la boîte d'outils Neptune ML.

L'environnement Conda reproduit avec précision l'environnement dans lequel votre modèle sera exécuté lorsque vous appellerez l'API d'entraînement de modèle. Tous les exemples de scripts d'entraînement et de transformation vous permettent de transmettre un indicateur `--local` de ligne de commande pour exécuter les scripts dans un environnement local afin de faciliter le débogage. Il

s'agit d'une bonne pratique lors du développement de votre propre modèle, car cela vous permet de tester l'implémentation de votre modèle de manière interactive et itérative. Lors de l'entraînement de modèle dans l'environnement d'entraînement de production Neptune ML, ce paramètre est omis.

Création d'un point de terminaison d'inférence à interroger

Un point de terminaison d'inférence vous permet d'interroger un modèle spécifique construit par le processus d'entraînement de modèle. Le point de terminaison se rattache au modèle le plus performant d'un type donné que le processus d'entraînement a pu générer. Le point de terminaison est alors en mesure d'accepter les requêtes Gremlin de Neptune et de renvoyer les prédictions de ce modèle pour les entrées dans les requêtes. Une fois que vous avez créé un point de terminaison d'inférence, celui-ci reste actif jusqu'à ce que vous le supprimiez.

Gestion des points de terminaison d'inférence pour Neptune ML

Une fois que vous avez terminé l'entraînement du modèle sur les données que vous avez exportées depuis Neptune, vous pouvez créer un point de terminaison d'inférence à l'aide d'une commande comme celle-ci :

AWS CLI

```
aws neptunedata create-ml-endpoint \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique ID for the new endpoint)" \
  --ml-model-training-job-id "(the model-training job-id of a completed job)"
```

Pour plus d'informations, consultez create-ml-endpointle manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.create_ml_endpoint(
    id='(a unique ID for the new endpoint)',
    mlModelTrainingJobId='(the model-training job-id of a completed job)'
)
```

```
print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/endpoints \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/endpoints \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Vous pouvez également créer un point de terminaison d'inférence à partir d'un modèle créé par une tâche de transformation de modèle terminée, à peu près de la même manière :

AWS CLI

```
aws neptunedata create-ml-endpoint \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique ID for the new endpoint)" \
  --ml-model-transform-job-id "(the model-transform job-id of a completed job)"
```

Pour plus d'informations, consultez create-ml-endpointle manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.create_ml_endpoint(
    id='(a unique ID for the new endpoint)',
    mlModelTransformJobId='(the model-transform job-id of a completed job)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/endpoints \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTransformJobId": "(the model-transform job-id of a completed job)"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
```

```
-X POST https://your-neptune-endpoint:port/ml/endpoints \
-H 'Content-Type: application/json' \
-d '{
      "id" : "(a unique ID for the new endpoint)",
      "mlModelTransformJobId": "(the model-transform job-id of a completed job)"
    }'
```

Les détails de l'utilisation de ces commandes sont fournis dans Commande endpoints, ainsi que des informations sur la façon d'obtenir le statut d'un point de terminaison, la façon de supprimer un point de terminaison et de répertorier tous les points de terminaison d'inférence.

Requêtes d'inférence dans Neptune ML

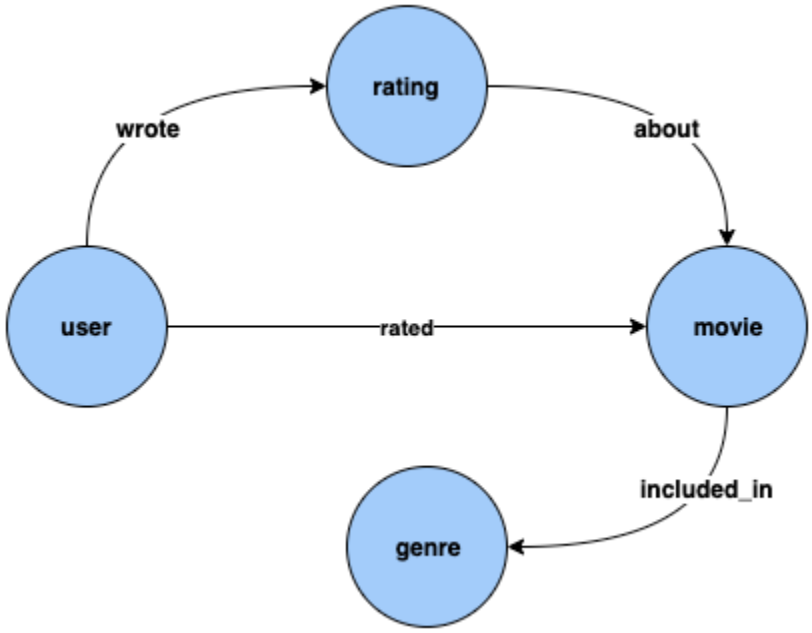
Vous pouvez utiliser Gremlin ou SPARQL pour interroger un point de terminaison d'inférence Neptune ML. Toutefois, l'inférence inductive en temps réel n'est actuellement prise en charge que pour les requêtes Gremlin.

Requêtes d'inférence Gremlin dans Neptune ML

Comme décrit dans Fonctionnalités de Neptune ML, Neptune ML prend en charge les modèles d'entraînement qui peuvent effectuer les types de tâches d'inférence suivants :

- Classification de nœud : prédit la fonctionnalité catégorielle d'une propriété de sommet.
- Régression de nœud : prédit une propriété numérique d'un sommet.
- Classification d'arête : prédit la fonctionnalité catégorielle d'une propriété d'arête.
- Régression d'arête : prédit une propriété numérique d'une arête.
- Prédiction des liens : prédit les nœuds de destination à partir d'un nœud source et d'une arête sortante, ou les nœuds sources à partir d'un nœud de destination et d'une arête entrante.

Nous pouvons illustrer ces différentes tâches à l'aide d'exemples utilisant le jeu de données de MovieLens 100 000 unités fourni par GroupLens Research. Ce jeu de données comprend des films, des utilisateurs et des évaluations de ces films par les utilisateurs, à partir desquels nous avons créé un graphe de propriétés comme celui-ci :



Classification de nœud : dans le jeu de données ci-dessus, `Genre` est un type de sommet connecté au type de sommet `Movie` par l'arête `included_in`. Toutefois, si nous modifions le jeu de données pour faire de `Genre` une fonctionnalité catégorielle du type de sommet `Movie`, le problème de la déduction de `Genre` pour les nouveaux films ajoutés au graphe de connaissances peut être résolu à l'aide de modèles de classification de nœud.

Régression de nœud : si nous considérons le type de sommet `Rating`, qui possède des propriétés telles que `timestamp` et `score`, le problème de l'inférence de la valeur numérique `Score` pour un élément `Rating` peut être résolu à l'aide de modèles de régression de nœud.

Classification des arêtes : De même, pour une `Rated` arête, si nous avons une propriété `Scale` qui peut avoir l'une des valeurs,,,,,`Love`,,`Like`,`Dislike`,, `NeutralHate`, alors le problème de la déduction de l'`Rated`arête `Scale` pour une nouvelle movies/ratings peut être résolu à l'aide de modèles de classification des arêtes.

Régression d'arête : de façon similaire, pour la même arête `Rated`, si une propriété `Score` contient une valeur numérique pour l'évaluation, celle-ci peut être déduite à partir des modèles de régression d'arête.

Prédiction des liens : des problèmes tels que la recherche des dix utilisateurs susceptibles d'évaluer le mieux un film donné ou la recherche des dix films qu'un utilisateur donné a le plus de chance d'évaluer relèvent de la prédiction des liens.

> **Note**
>
> Pour les cas d'utilisation de Neptune ML, nous disposons d'un ensemble très complet de blocs-notes conçus pour favoriser la compréhension pratique de chaque cas d'utilisation. Vous pouvez créer ces blocs-notes avec votre cluster Neptune en utilisant le modèle Neptune CloudFormation ML pour créer un cluster Neptune ML. Ces blocs-notes sont également disponibles sur github.

Rubriques

- Prédicats Neptune ML utilisés dans les requêtes d'inférence Gremlin
- Requêtes de classification de nœud Gremlin dans Neptune ML
- Requêtes de régression de nœud Gremlin dans Neptune ML
- Requêtes de classification d'arête Gremlin dans Neptune ML
- Requêtes de régression d'arête Gremlin dans Neptune ML
- Requêtes de prédiction de lien Gremlin utilisant des modèles de prédiction de lien dans Neptune ML
- Liste des exceptions pour les requêtes d'inférence Gremlin de Neptune ML

Prédicats Neptune ML utilisés dans les requêtes d'inférence Gremlin

`Neptune#ml.deterministic`

Ce prédicat est une option pour les requêtes d'inférence inductive, c'est-à-dire pour les requêtes qui incluent le prédicat `Neptune#ml.inductiveInference`.

Lors de l'utilisation de l'inférence inductive, le moteur Neptune crée le sous-graphe approprié pour évaluer le modèle GNN entraîné, et les exigences de ce sous-graphe dépendent des paramètres du modèle final. Plus précisément, le paramètre `num-layer` détermine le nombre de sauts de traversée depuis les arêtes ou les nœuds cibles, et le paramètre `fanouts` spécifie le nombre de voisins à échantillonner à chaque saut (voir Paramètres HPO).

Par défaut, les requêtes d'inférence inductive s'exécutent en mode non déterministe, dans lequel Neptune construit le voisinage de manière aléatoire. Lors de la réalisation de prédictions, cet échantillonnage normal de voisins aléatoires fournit parfois des prédictions différentes.

Lorsque vous incluez `Neptune#ml.deterministic` dans une requête d'inférence inductive, le moteur Neptune tente d'échantillonner les voisins de manière déterministe afin que plusieurs invocations de la même requête renvoient les mêmes résultats à chaque fois. Il n'est toutefois pas possible de garantir des résultats totalement déterministes, car les modifications apportées au graphe sous-jacent et les artefacts des systèmes distribués peuvent toujours introduire des fluctuations.

Vous incluez le prédicat `Neptune#ml.deterministic` dans une requête comme suit :

```
.with("Neptune#ml.deterministic")
```

Si le prédicat `Neptune#ml.deterministic` est inclus dans une requête qui n'inclut pas également `Neptune#ml.inductiveInference`, il est tout simplement ignoré.

`Neptune#ml.disableInductiveInferenceMetadataCache`

Ce prédicat est une option pour les requêtes d'inférence inductive, c'est-à-dire pour les requêtes qui incluent le prédicat `Neptune#ml.inductiveInference`.

Pour les requêtes d'inférence inductive, Neptune utilise un fichier de métadonnées stocké dans Amazon S3 pour décider du nombre de sauts et de la diffusion en éventail lors de la construction du voisinage. Neptune met normalement en cache les métadonnées de ce modèle pour éviter de récupérer le fichier à plusieurs reprises depuis Amazon S3. La mise en cache peut être désactivée en incluant le prédicat `Neptune#ml.disableInductiveInferenceMetadataCache` dans la requête. Bien que Neptune puisse être plus lent à récupérer les métadonnées directement depuis

Amazon S3, cela est utile lorsque le point de terminaison SageMaker AI a été mis à jour après un réentraînement ou une transformation et que le cache est périmé.

Vous incluez le prédicat `Neptune#ml.disableInductiveInferenceMetadataCache` dans une requête comme suit :

```
.with("Neptune#ml.disableInductiveInferenceMetadataCache")
```

Voici à quoi peut ressembler un exemple de requête dans un bloc-notes Jupyter :

```
%%gremlin
g.with("Neptune#ml.endpoint", "ep1")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .with("Neptune#ml.disableInductiveInferenceMetadataCache")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
```

`Neptune#ml.endpoint`

Le prédicat `Neptune#ml.endpoint` est utilisé dans une étape `with()` pour spécifier le point de terminaison d'inférence, si nécessaire :

```
 .with("Neptune#ml.endpoint", "the model's SageMaker AI inference endpoint")
```

Vous pouvez identifier le point de terminaison par son `id` ou son URL. Par exemple :

```
 .with( "Neptune#ml.endpoint", "node-classification-movie-lens-endpoint" )
```

Ou:

```
 .with( "Neptune#ml.endpoint", "https://runtime.sagemaker.us-east-1.amazonaws.com/
endpoints/node-classification-movie-lens-endpoint/invocations" )
```

> **Note**
>
> Si vous définissez le paramètre `neptune_ml_endpoint` dans votre groupe de paramètres de cluster de bases de données Neptune sur l'`id` ou l'URL du point de terminaison, vous n'avez pas besoin d'inclure le prédicat `Neptune#ml.endpoint` dans chaque requête.

Neptune#ml.iamRoleArn

`Neptune#ml.iamRoleArn`est utilisé dans une `with()` étape pour spécifier l'ARN du rôle IAM d'exécution de l' SageMaker IA, si nécessaire :

```
.with("Neptune#ml.iamRoleArn", "the ARN for the SageMaker AI execution IAM role")
```

Pour plus d'informations sur la création du rôle IAM d'exécution de l' SageMaker IA, consultezCréation d'un NeptuneSageMaker IAMRole rôle personnalisé.

> **Note**
>
> Si vous définissez le `neptune_ml_iam_role` paramètre dans le groupe de paramètres de votre cluster de base de données Neptune sur l'ARN de votre rôle IAM d'exécution SageMaker AI, vous n'avez pas besoin d'inclure le `Neptune#ml.iamRoleArn` prédicat dans chaque requête.

Neptune#ml.inductiveInference

L'inférence transductive est activée par défaut dans Gremlin. Pour effectuer une requête d'inférence inductive en temps réel, incluez le prédicat `Neptune#ml.inductiveInference` comme suit :

```
.with("Neptune#ml.inductiveInference")
```

Si votre graphe est dynamique, l'inférence inductive est souvent le meilleur choix, mais si votre graphe est statique, l'inférence transductive est plus rapide et plus efficace.

Neptune#ml.limit

Le prédicat `Neptune#ml.limit` limite éventuellement le nombre de résultats renvoyés par entité :

```
.with( "Neptune#ml.limit", 2 )
```

Par défaut, la limite est de 1 et le nombre maximal pouvant être défini est de 100.

Neptune#ml.threshold

Le prédicat `Neptune#ml.threshold` établit éventuellement un seuil limite pour les scores de résultat :

```
.with( "Neptune#ml.threshold", 0.5D )
```

Cela vous permet de rejeter tous les résultats dont les scores sont inférieurs au seuil spécifié.

Neptune#ml.classification

Le `Neptune#ml.classification` prédicat est attaché à l'`properties()`étape visant à établir que les propriétés doivent être extraites du point de terminaison SageMaker AI du modèle de classification des nœuds :

```
.properties( "property key of the node classification
model" ).with( "Neptune#ml.classification" )
```

Neptune#ml.regression

Le `Neptune#ml.regression` prédicat est attaché à l'`properties()`étape visant à établir que les propriétés doivent être extraites du point de terminaison SageMaker AI du modèle de régression des nœuds :

```
.properties( "property key of the node regression
model" ).with( "Neptune#ml.regression" )
```

Neptune#ml.prediction

Le prédicat `Neptune#ml.prediction` est attaché aux étapes `in()` et `out()` pour établir qu'il s'agit d'une requête de prédiction de lien :

```
.in("edge label of the link prediction
model").with("Neptune#ml.prediction").hasLabel("target node label")
```

Neptune#ml.score

Le prédicat `Neptune#ml.score` est utilisé dans les requêtes de classification de nœud ou d'arête de Gremlin pour extraire un score de confiance de machine learning. Le prédicat `Neptune#ml.score` doit être transmis avec le prédicat de requête à l'étape `properties()` afin d'obtenir un score de confiance ML pour les requêtes de classification de nœud ou d'arête.

Vous pouvez trouver un exemple de classification de nœud avec d'autres exemples de classification de nœud, et un exemple de classification d'arête dans la section Classification d'arête.

Requêtes de classification de nœud Gremlin dans Neptune ML

Pour la classification de nœud Gremlin dans Neptune ML :

- Le modèle est entraîné sur une seule propriété des sommets. L'ensemble des valeurs uniques de cette propriété est appelé « ensemble de classes de nœuds », ou simplement « classes ».
- La classe de nœuds ou la valeur de propriété catégorielle de la propriété d'un sommet peut être déduite du modèle de classification de nœud. Ceci est utile lorsque cette propriété n'est pas déjà attachée au sommet.
- Pour extraire une ou plusieurs classes d'un modèle de classification de nœud, vous devez utiliser l'étape `with()` avec le prédicat `Neptune#ml.classification` pour configurer l'étape `properties()`. Le format de sortie est similaire à ce à quoi vous vous attendriez s'il s'agissait de propriétés de sommet.

> **Note**
>
> La classification de nœud ne fonctionne qu'avec les valeurs de propriété de type string. Cela signifie que des valeurs de propriété numériques telles que `0` ou `1` ne sont pas prises en charge, alors que les valeurs équivalentes de type string `"0"` et `"1"` le sont. De même, les valeurs de propriété booléenne `true` et `false` ne fonctionnent pas, mais `"true"` et `"false"` fonctionnent.

Voici un exemple de requête de classification de nœud :

```
g.with( "Neptune#ml.endpoint","node-classification-movie-lens-endpoint" )
 .with( "Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role" )
 .with( "Neptune#ml.limit", 2 )
 .with( "Neptune#ml.threshold", 0.5D )
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre").with("Neptune#ml.classification")
```

La sortie de cette requête ressemble à ce qui suit :

```
==>vp[genre->Action]
==>vp[genre->Crime]
==>vp[genre->Comedy]
```

Dans la requête ci-dessus, les étapes `V()` et `properties()` sont utilisées comme suit :

L'étape `V()` contient l'ensemble de sommets pour lequel vous souhaitez extraire les classes du modèle de classification de nœud :

```
  .V( "movie_1", "movie_2", "movie_3" )
```

L'étape `properties()` contient la clé sur laquelle le modèle a été entraîné et contient `.with("Neptune#ml.classification")` pour indiquer qu'il s'agit d'une requête d'inférence ML de classification de nœud.

Les clés de propriété multiples ne sont actuellement pas prises en charge dans une étape `properties().with("Neptune#ml.classification")`. Par exemple, la requête suivante lève une exception :

```
g.with("Neptune#ml.endpoint", "node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre", "other_label").with("Neptune#ml.classification")
```

Pour le message d'erreur spécifique, consultez la liste des exceptions de Neptune ML.

Une étape `properties().with("Neptune#ml.classification")` peut être utilisée en combinaison avec l'une quelconque des étapes suivantes :

- `value()`
- `value().is()`
- `hasValue()`
- `has(value,"")`
- `key()`
- `key().is()`
- `hasKey()`
- `has(key,"")`
- `path()`

Autres requêtes de classification de nœud

Si le point de terminaison d'inférence et le rôle IAM correspondant ont tous les deux été enregistrés dans le groupe de paramètres de votre cluster de bases de données, une requête de classification de nœud peut être aussi simple que ceci :

```
g.V("movie_1", "movie_2",
 "movie_3").properties("genre").with("Neptune#ml.classification")
```

Vous pouvez mélanger des propriétés et des classes de sommets dans une requête en utilisant l'étape `union()` :

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .union(
   properties("genre").with("Neptune#ml.classification"),
   properties("genre")
 )
```

Vous pouvez également effectuer une requête illimitée telle que celle-ci :

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V()
 .properties("genre").with("Neptune#ml.classification")
```

Vous pouvez récupérer les classes de nœuds ainsi que les sommets en utilisant l'étape `select()` associée à l'étape `as()` :

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" ).as("vertex")
 .properties("genre").with("Neptune#ml.classification").as("properties")
 .select("vertex","properties")
```

Vous pouvez également filtrer les classes de nœuds, comme illustré dans ces exemples :

```
g.with("Neptune#ml.endpoint", "node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
```

```
  .properties("genre").with("Neptune#ml.classification")
  .has(value, "Horror")

g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V( "movie_1", "movie_2", "movie_3" )
  .properties("genre").with("Neptune#ml.classification")
  .has(value, P.eq("Action"))

g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V( "movie_1", "movie_2", "movie_3" )
  .properties("genre").with("Neptune#ml.classification")
  .has(value, P.within("Action", "Horror"))
```

Vous pouvez obtenir un score de confiance pour la classification de nœud en utilisant le prédicat `Neptune#ml.score` :

```
 g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V( "movie_1", "movie_2", "movie_3" )
  .properties("genre", "Neptune#ml.score").with("Neptune#ml.classification")
```

La réponse ressemble alors à ceci :

```
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.01234567]
==>vp[genre->Crime]
==>vp[Neptune#ml.score->0.543210]
==>vp[genre->Comedy]
==>vp[Neptune#ml.score->0.10101]
```

Utilisation de l'inférence inductive dans une requête de classification de nœud

Supposons que vous ajoutiez un nouveau nœud à un graphe existant, dans un bloc-notes Jupyter, comme suit :

```
%%gremlin
g.addV('label1').property(id,'101').as('newV')
  .V('1').as('oldV1')
  .V('2').as('oldV2')
  .addE('eLabel1').from('newV').to('oldV1')
```

```
  .addE('eLabel2').from('oldV2').to('newV')
```

Vous pouvez ensuite utiliser une requête d'inférence inductive pour obtenir un genre et un score de confiance reflétant le nouveau nœud :

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("genre", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
```

Toutefois, si vous exécutez la requête plusieurs fois, vous pouvez obtenir des résultats quelque peu différents :

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.21365921]
```

Vous pouvez rendre la même requête déterministe :

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("genre", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Dans ce cas, les résultats sont à peu près les mêmes à chaque fois :

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
```

Requêtes de régression de nœud Gremlin dans Neptune ML

La régression de nœud est similaire à la classification de nœud, si ce n'est que la valeur déduite du modèle de régression pour chaque nœud est numérique. Vous pouvez utiliser les mêmes requêtes Gremlin pour la régression de nœud que pour la classification de nœud, à l'exception des différences suivantes :

- Encore une fois, dans Neptune ML, les nœuds font référence à des sommets.
- L'étape `properties()` prend la forme `properties().with("Neptune#ml.regression")` à la place de `properties().with("Neptune#ml.classification")`.
- Les prédicats "`Neptune#ml.limit`" et "`Neptune#ml.threshold`" ne sont pas applicables.
- Quand vous filtrez sur la valeur, vous devez spécifier une valeur numérique.

Voici un exemple de requête de classification de sommet :

```
g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1","movie_2","movie_3")
 .properties("revenue").with("Neptune#ml.regression")
```

Vous pouvez filtrer sur la valeur déduite à l'aide d'un modèle de régression, comme illustré dans les exemples suivants :

```
g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1","movie_2","movie_3")
 .properties("revenue").with("Neptune#ml.regression")
 .value().is(P.gte(1600000))

g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1","movie_2","movie_3")
 .properties("revenue").with("Neptune#ml.regression")
 .hasValue(P.lte(1600000D))
```

Utilisation de l'inférence inductive dans une requête de régression de nœud

Supposons que vous ajoutiez un nouveau nœud à un graphe existant, dans un bloc-notes Jupyter, comme suit :

```
%%gremlin
g.addV('label1').property(id,'101').as('newV')
 .V('1').as('oldV1')
 .V('2').as('oldV2')
 .addE('eLabel1').from('newV').to('oldV1')
 .addE('eLabel2').from('oldV2').to('newV')
```

Vous pouvez ensuite utiliser une requête d'inférence inductive pour obtenir une évaluation prenant en compte le nouveau nœud :

```
%%gremlin
g.with("Neptune#ml.endpoint", "nr-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
```

La requête n'étant pas déterministe, elle peut renvoyer des résultats légèrement différents si vous l'exécutez plusieurs fois, en fonction du voisinage :

```
# First time
==>vp[rating->9.1]

# Second time
==>vp[rating->8.9]
```

Si vous avez besoin de résultats plus cohérents, vous pouvez rendre la requête déterministe :

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Désormais, les résultats sont quasiment les mêmes à chaque fois :

```
# First time
==>vp[rating->9.1]
```

```
# Second time
==>vp[rating->9.1]
```

Requêtes de classification d'arête Gremlin dans Neptune ML

Pour la classification d'arête Gremlin dans Neptune ML :

- Le modèle est entraîné sur une seule propriété des arêtes. L'ensemble des valeurs uniques de cette propriété est appelé « ensemble de classes ».
- La classe ou la valeur de propriété catégorielle d'une arête peut être déduite du modèle de classification d'arête, ce qui est utile quand cette propriété n'est pas encore attachée à l'arête.
- Pour extraire une ou plusieurs classes d'un modèle de classification d'arête, vous devez utiliser l'étape `with()` avec le prédicat `"Neptune#ml.classification"` pour configurer l'étape `properties()`. Le format de sortie est similaire à celui auquel vous vous attendriez s'il s'agissait de propriétés d'arête.

> **Note**
>
> La classification d'arête ne fonctionne qu'avec les valeurs de propriété de type string. Cela signifie que des valeurs de propriété numériques telles que `0` ou `1` ne sont pas prises en charge, alors que les valeurs équivalentes de type string `"0"` et `"1"` le sont. De même, les valeurs de propriété booléenne `true` et `false` ne fonctionnent pas, mais `"true"` et `"false"` fonctionnent.

Voici un exemple de requête de classification d'arête qui demande un score de confiance en utilisant le prédicat `Neptune#ml.score` :

```
g.with("Neptune#ml.endpoint","edge-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by", "Neptune#ml.score").with("Neptune#ml.classification")
```

La réponse ressemble alors à ceci :

```
==>p[knows_by->"Family"]
==>p[Neptune#ml.score->0.01234567]
==>p[knows_by->"Friends"]
```

```
==>p[Neptune#ml.score->0.543210]
==>p[knows_by->"Colleagues"]
==>p[Neptune#ml.score->0.10101]
```

Syntaxe d'une requête de classification d'arête Gremlin

Pour un graphe simple où `User` représente le nœud de tête et le nœud de queue, et où `Relationship` est l'arête qui les relie, voici un exemple de requête de classification d'arête :

```
g.with("Neptune#ml.endpoint","edge-classification-social-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by").with("Neptune#ml.classification")
```

La sortie de cette requête ressemble à ce qui suit :

```
==>p[knows_by->"Family"]
==>p[knows_by->"Friends"]
==>p[knows_by->"Colleagues"]
```

Dans la requête ci-dessus, les étapes `E()` et `properties()` sont utilisées comme suit :

- L'étape `E()` contient l'ensemble des arêtes pour lesquelles vous souhaitez extraire les classes du modèle de classification d'arête :

  ```
  .E("relationship_1","relationship_2","relationship_3")
  ```

- L'étape `properties()` contient la clé sur laquelle le modèle a été entraîné et contient `.with("Neptune#ml.classification")` pour indiquer qu'il s'agit d'une requête d'inférence ML de classification d'arête.

Les clés de propriété multiples ne sont actuellement pas prises en charge dans une étape `properties().with("Neptune#ml.classification")`. Par exemple, la requête suivante lève une exception :

```
g.with("Neptune#ml.endpoint","edge-classification-social-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by", "other_label").with("Neptune#ml.classification")
```

Pour des messages d'erreur spécifiques, consultez Liste des exceptions pour les requêtes d'inférence Gremlin de Neptune ML.

Une étape `properties().with("Neptune#ml.classification")` peut être utilisée en combinaison avec l'une quelconque des étapes suivantes :

- `value()`
- `value().is()`
- `hasValue()`
- `has(value,"")`
- `key()`
- `key().is()`
- `hasKey()`
- `has(key,"")`
- `path()`

Utilisation de l'inférence inductive dans une requête de classification d'arête

Supposons que vous ajoutiez une nouvelle arête à un graphe existant, dans un bloc-notes Jupyter, comme suit :

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

Vous pouvez ensuite utiliser une requête d'inférence inductive pour obtenir une échelle prenant en compte la nouvelle arête :

```
%%gremlin
g.with("Neptune#ml.endpoint", "ec-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("scale", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
```

La requête n'étant pas déterministe, elle peut renvoyer des résultats légèrement différents si vous l'exécutez plusieurs fois, en fonction du voisinage aléatoire :

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.21365921]
```

Si vous avez besoin de résultats plus cohérents, vous pouvez rendre la requête déterministe :

```
%%gremlin
g.with("Neptune#ml.endpoint", "ec-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("scale", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Désormais, les résultats sont plus ou moins les mêmes chaque fois que vous exécutez la requête :

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]
```

Requêtes de régression d'arête Gremlin dans Neptune ML

La régression d'arête est similaire à la classification d'arête, si ce n'est que la valeur déduite du modèle ML est numérique. Pour la régression d'arête, Neptune ML prend en charge les mêmes requêtes que pour la classification.

Les principaux points à noter sont les suivants :

- Vous devez utiliser le prédicat ML `"Neptune#ml.regression"` pour configurer l'étape `properties()` correspondant à ce cas d'utilisation.
- Les prédicats `"Neptune#ml.limit"` et `"Neptune#ml.threshold"` ne sont pas applicables dans ce cas d'utilisation.

- Pour filtrer sur la valeur, vous devez spécifier la valeur sous forme numérique.

Syntaxe d'une requête de régression d'arête Gremlin

Pour un graphe simple où `User` est le nœud de tête, `Movie` le nœud de queue et `Rated` l'arête qui les relie, voici un exemple de requête de régression d'arête qui recherche la valeur d'évaluation numérique, appelée score ici, pour l'arête `Rated` :

```
g.with("Neptune#ml.endpoint","edge-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("rating_1","rating_2","rating_3")
 .properties("score").with("Neptune#ml.regression")
```

Vous pouvez également filtrer sur une valeur déduite du modèle de régression ML. Pour les arêtes `Rated` existantes (de `User` à `Movie`) identifiées par `"rating_1"`, `"rating_2"` et `"rating_3"`, pour lesquelles la propriété d'arête `Score` n'est pas présente pour ces notations, vous pouvez utiliser une requête comme la suivante pour déduire `Score` pour les arêtes où elle est supérieure ou égale à 9 :

```
g.with("Neptune#ml.endpoint","edge-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("rating_1","rating_2","rating_3")
 .properties("score").with("Neptune#ml.regression")
 .value().is(P.gte(9))
```

Utilisation de l'inférence inductive dans une requête de régression d'arête

Supposons que vous ajoutiez une nouvelle arête à un graphe existant, dans un bloc-notes Jupyter, comme suit :

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

Vous pouvez ensuite utiliser une requête d'inférence inductive pour obtenir un score prenant en compte la nouvelle arête :

```
%%gremlin
g.with("Neptune#ml.endpoint", "er-ep")
```

```
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("score")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
```

La requête n'étant pas déterministe, elle peut renvoyer des résultats légèrement différents si vous l'exécutez plusieurs fois, en fonction du voisinage aléatoire :

```
# First time
==>ep[score->96]

# Second time
==>ep[score->91]
```

Si vous avez besoin de résultats plus cohérents, vous pouvez rendre la requête déterministe :

```
%%gremlin
g.with("Neptune#ml.endpoint", "er-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("score")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Désormais, les résultats sont plus ou moins les mêmes chaque fois que vous exécutez la requête :

```
# First time
==>ep[score->96]

# Second time
==>ep[score->96]
```

Requêtes de prédiction de lien Gremlin utilisant des modèles de prédiction de lien dans Neptune ML

Les modèles de prédiction de lien peuvent résoudre des problèmes tels que les suivants :

- Prédiction de nœud de tête : étant donné un sommet et un type d'arête, à partir de quels sommets est-il probable que ce sommet soit lié ?
- Prédiction de nœud de queue : étant donné un sommet et une étiquette d'arête, à quels sommets est-il probable que ce sommet soit lié ?

> **Note**
>
> La prédiction d'arête n'est pas encore prise en charge dans Neptune ML.

Pour les exemples ci-dessous, considérez un graphe simple avec les sommets `User` et `Movie` liés par l'arête `Rated`.

Voici un exemple de requête de prédiction de nœud de tête, utilisé pour prédire les cinq utilisateurs qui ont le plus de chance d'évaluer les films `"movie_1"`, `"movie_2"` et `"movie_3"` :

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .with("Neptune#ml.limit", 5)
 .V("movie_1", "movie_2", "movie_3")
 .in("rated").with("Neptune#ml.prediction").hasLabel("user")
```

Voici un exemple similaire pour la prédiction de nœud de queue, utilisé pour prédire les cinq films qu'il est le plus probable que l'utilisateur `"user_1"` évalue :

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out("rated").with("Neptune#ml.prediction").hasLabel("movie")
```

L'étiquette d'arête et l'étiquette de sommet prédit sont toutes deux obligatoires. Si l'une d'elles est omise, une exception est levée. Par exemple, la requête suivante sans étiquette de sommet prédit lève une exception :

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out("rated").with("Neptune#ml.prediction")
```

De façon similaire, la requête suivante sans étiquette d'arête lève une exception :

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out().with("Neptune#ml.prediction").hasLabel("movie")
```

Pour connaître les messages d'erreur spécifiques renvoyés par ces exceptions, consultez la liste des exceptions de Neptune ML.

Autres requêtes de prédiction de lien

Vous pouvez utiliser l'étape `select()` avec l'étape `as()` pour générer les sommets prédits avec les sommets en entrée :

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1").as("source")
 .in("rated").with("Neptune#ml.prediction").hasLabel("user").as("target")
 .select("source","target")

g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1").as("source")
 .out("rated").with("Neptune#ml.prediction").hasLabel("movie").as("target")
 .select("source","target")
```

Vous pouvez effectuer des requêtes illimitées, comme celles-ci :

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out("rated").with("Neptune#ml.prediction").hasLabel("movie")

g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1")
 .in("rated").with("Neptune#ml.prediction").hasLabel("user")
```

Utilisation de l'inférence inductive dans une requête de prédiction de lien

Supposons que vous ajoutiez un nouveau nœud à un graphe existant, dans un bloc-notes Jupyter, comme suit :

```
%%gremlin
g.addV('label1').property(id,'101').as('newV1')
 .addV('label2').property(id,'102').as('newV2')
 .V('1').as('oldV1')
 .V('2').as('oldV2')
```

```
  .addE('eLabel1').from('newV1').to('oldV1')
  .addE('eLabel2').from('oldV2').to('newV2')
```

Vous pouvez alors utiliser une requête d'inférence inductive pour prédire le nœud de tête, en tenant compte du nouveau nœud :

```
%%gremlin
g.with("Neptune#ml.endpoint", "lp-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').out("eLabel1")
 .with("Neptune#ml.prediction")
 .with("Neptune#ml.inductiveInference")
 .hasLabel("label2")
```

Résultat :

```
==>V[2]
```

De façon similaire, vous pouvez utiliser une requête d'inférence inductive pour prédire le nœud de queue, en tenant compte du nouveau nœud :

```
%%gremlin
g.with("Neptune#ml.endpoint", "lp-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('102').in("eLabel2")
 .with("Neptune#ml.prediction")
 .with("Neptune#ml.inductiveInference")
 .hasLabel("label1")
```

Résultat :

```
==>V[1]
```

Liste des exceptions pour les requêtes d'inférence Gremlin de Neptune ML

Il s'agit d'une liste complète des exceptions qui peuvent se produire lors de l'exécution de requêtes d'inférence Neptune ML Gremlin. Ces exceptions couvrent toute une série de problèmes, tels que les problèmes liés au rôle ou au point de terminaison IAM spécifié, aux étapes Gremlin non prises en charge et aux limitations du nombre de requêtes d'inférence ML par requête. Chaque entrée inclut un message détaillé décrivant le problème.

- **`BadRequestException`** : les informations d'identification pour le rôle fourni ne peuvent pas être chargées.

  Message : `Unable to load credentials for role: the specified IAM Role ARN.`

- **`BadRequestException`**— Le rôle IAM spécifié n'est pas autorisé à invoquer le point de terminaison SageMaker AI.

  Message : `User: the specified IAM Role ARN is not authorized to perform: sagemaker:InvokeEndpoint on resource: the specified endpoint.`

- **`BadRequestException`** : le point de terminaison spécifié n'existe pas.

  Message : `Endpoint the specified endpoint not found.`

- **`InternalFailureException`** : impossible d'extraire les métadonnées d'inférence inductive en temps réel de Neptune ML depuis Amazon S3.

  Message : `Unable to fetch Neptune ML - Real-Time Inductive Inference metadata from S3. Check the permissions of the S3 bucket or if the Neptune instance can connect to S3.`

- **`InternalFailureException`** : Neptune ML ne trouve pas le fichier de métadonnées pour l'inférence inductive en temps réel dans Amazon S3.

  Message : `Neptune ML cannot find the metadata file for Real-Time Inductive Inference in S3.`

- **`InvalidParameterException`** : le point de terminaison spécifié n'est pas valide syntaxiquement.

  Message : `Invalid endpoint provided for external service query.`

- **`InvalidParameterException`**— L'ARN du rôle IAM SageMaker d'exécution spécifié n'est pas syntaxiquement valide.

  Message : `Invalid IAM role ARN provided for external service query.`

- **`InvalidParameterException`** : plusieurs clés de propriété sont spécifiées à l'étape `properties()` d'une requête.

  Message : `ML inference queries are currently supported for one property key.`

- **`InvalidParameterException`** : plusieurs étiquettes d'arête sont spécifiées dans une requête.

Message : `ML inference are currently supported only with one edge label.`

- **`InvalidParameterException`** : plusieurs contraintes d'étiquette de sommet sont spécifiées dans une requête.

  Message : `ML inference are currently supported only with one vertex label constraint.`

- **`InvalidParameterException`** : les deux prédicats `Neptune#ml.classification` et `Neptune#ml.regression` sont présents dans la même requête.

  Message : `Both regression and classification ML predicates cannot be specified in the query.`

- **`InvalidParameterException`** : plusieurs étiquettes d'arête ont été spécifiées à l'étape `in()` ou `out()` dans une requête de prédiction de lien.

  Message : `ML inference are currently supported only with one edge label.`

- **`InvalidParameterException`** : plusieurs clés de propriété ont été spécifiées avec Neptune#ml.score.

  Message : `Neptune ML inference queries are currently supported for one property key and one Neptune#ml.score property key.`

- **`MissingParameterException`** : le point de terminaison n'a pas été spécifié dans la requête ou en tant que paramètre de cluster de bases de données.

  Message : `No endpoint provided for external service query.`

- MissingParameterException— Le rôle IAM d'exécution de l' SageMaker IA n'a pas été spécifié dans la requête ni en tant que paramètre de cluster de base de données.

  Message : `No IAM role ARN provided for external service query.`

- **`MissingParameterException`** : la clé de propriété est absente de l'étape `properties()` d'une requête.

  Message : `Property key needs to be specified using properties() step for ML inference queries.`

- **`MissingParameterException`** : aucune étiquette d'arête n'a été spécifiée à l'étape `in()` ou `out()` d'une requête de prédiction de lien.

Message : `Edge label needs to be specified while using in() or out() step for ML inference queries.`

- **`MissingParameterException`** : aucune clé de propriété n'a été spécifiée avec Neptune#ml.score.

  Message : `Property key needs to be specified along with Neptune#ml.score property key while using the properties() step for Neptune ML inference queries.`

- **`UnsupportedOperationException`** : l'étape `both()` est utilisée dans une requête de prédiction de lien.

  Message : `ML inference queries are currently not supported with both() step.`

- **`UnsupportedOperationException`** : aucune étiquette de sommet prédit n'a été spécifiée à l'étape `has()` avec l'étape `in()` ou `out()` dans une requête de prédiction de lien.

  Message : `Predicted vertex label needs to be specified using has() step for ML inference queries.`

- **`UnsupportedOperationException`** : les requêtes d'inférence inductive ML Gremlin ne sont actuellement pas prises en charge avec des étapes non optimisées.

  Message : `Neptune ML - Real-Time Inductive Inference queries are currently not supported with Gremlin steps which are not optimized for Neptune. Check the Neptune User Guide for a list of Neptune-optimized steps.`

- **`UnsupportedOperationException`** : les requêtes d'inférence Neptune ML ne sont actuellement pas prises en charge dans une étape `repeat`.

  Message : `Neptune ML inference queries are currently not supported inside a repeat step.`

- **`UnsupportedOperationException`** : pas plus d'une seule requête d'inférence Neptune ML n'est actuellement prise en charge par requête Gremlin.

  Message : `Neptune ML inference queries are currently supported only with one ML inference query per gremlin query.`

Requêtes d'inférence SPARQL dans Neptune ML

Neptune ML mappe le graphe RDF dans un graphe de propriétés pour modéliser la tâche ML. Actuellement, il prend en charge les cas d'utilisation suivants :

- Classification d'objet : prédit la fonctionnalité catégorielle d'un objet.
- Régression d'objet : prédit une propriété numérique d'un objet.
- Prédiction d'objet : prédit un objet en fonction d'un sujet et d'une relation.
- Prédiction de sujet : prédit un sujet en fonction d'un objet et d'une relation.

> **Note**
>
> Neptune ML ne prend pas en charge les cas d'utilisation de régression et de classification de sujet avec SPARQL.

Prédicats Neptune ML utilisés dans les requêtes d'inférence SPARQL

Les prédicats suivants sont utilisés avec l'inférence SPARQL :

Prédicat **`neptune-ml:timeout`**

Spécifie le délai d'expiration de la connexion avec le serveur distant. À ne pas confondre avec le délai d'expiration de la demande de requête, qui est le temps maximal que le serveur peut mettre pour satisfaire une demande.

Notez que si le délai d'expiration de requête survient avant le délai d'expiration du service spécifié par le prédicat `neptune-ml:timeout`, la connexion au service est également annulée.

Prédicat **`neptune-ml:outputClass`**

Le prédicat `neptune-ml:outputClass` est uniquement utilisé pour définir la classe de l'objet prédit pour la prédiction d'objet ou du sujet prédit pour la prédiction de sujet.

Prédicat **`neptune-ml:outputScore`**

Le prédicat `neptune-ml:outputScore` est un nombre positif qui représente la probabilité que la sortie d'un modèle de machine learning soit correcte.

Prédicat **neptune-ml:modelType**

Le prédicat `neptune-ml:modelType` indique le type de modèle de machine learning en cours d'entraînement :

- `OBJECT_CLASSIFICATION`
- `OBJECT_REGRESSION`
- `OBJECT_PREDICTION`
- `SUBJECT_PREDICTION`

Prédicat **neptune-ml:input**

Le `neptune-ml:input` prédicat fait référence à la liste des entrées URIs utilisées pour Neptune ML.

Prédicat **neptune-ml:output**

Le prédicat `neptune-ml:output` fait référence à la liste des ensembles de liaisons où Neptune ML renvoie les résultats.

Prédicat **neptune-ml:predicate**

Le prédicat `neptune-ml:predicate` est utilisé différemment en fonction de la tâche exécutée :

- Pour une prédiction d'objet ou de sujet : définit le type de prédicat (type d'arête ou de relation).
- Pour une classification et régression d'objet : définit le littéral (propriété) que nous voulons prédire.

Prédicat **neptune-ml:batchSize**

`neptune-ml:batchSize` spécifie la taille d'entrée pour l'appel de service à distance.

Exemples de classification d'objet SPARQL

Pour une classification d'objet SPARQL dans Neptune ML, le modèle est entraîné sur l'une des valeurs de prédicat. Ceci est utile lorsque ce prédicat n'est pas déjà présent avec un sujet donné.

Seules les valeurs de prédicat catégorielles peuvent être déduites à l'aide du modèle de classification d'objet.

La requête suivante cherche à prédire la valeur de prédicat <http://www.example.org/team> pour toutes les entrées de type `foaf:Person` :

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_CLASSIFICATION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/team> ;
                      neptune-ml:output ?output .
  }
}
```

Cette requête peut être personnalisée comme suit :

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-account-balance-endpoint' ;
                      neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                      neptune-ml:batchSize "40"^^xsd:integer ;
                      neptune-ml:timeout "1000"^^xsd:integer ;

                      neptune-ml:modelType 'OBJECT_CLASSIFICATION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/team> ;
                      neptune-ml:output ?output .
  }
}
```

Exemples de régression d'objet SPARQL

La régression d'objet est similaire à la classification d'objet, si ce n'est qu'une valeur de prédicat numérique est déduite du modèle de régression pour chaque nœud. Vous pouvez utiliser les mêmes requêtes SPARQL pour la régression d'objet que pour la classification d'objet, à l'exception du fait que les prédicats `the Neptune#ml.limit` et `Neptune#ml.threshold` ne sont pas applicables.

La requête suivante cherche à prédire la valeur de prédicat <http://www.example.org/accountbalance> pour toutes les entrées de type `foaf:Person` :

```
SELECT * WHERE { ?input a foaf:Person .
```

```
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_REGRESSION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/accountbalance> ;
                      neptune-ml:output ?output .
  }
}
```

Cette requête peut être personnalisée comme suit :

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-account-balance-endpoint' ;
                      neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                      neptune-ml:batchSize "40"^^xsd:integer ;
                      neptune-ml:timeout "1000"^^xsd:integer ;

                      neptune-ml:modelType 'OBJECT_REGRESSION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/accountbalance> ;
                      neptune-ml:output ?output .
  }
}
```

Exemple de prédiction d'objet SPARQL

La prédiction d'objet prédit la valeur de l'objet pour un sujet et un prédicat donnés.

La requête de prédiction d'objet suivante cherche à prédire quel film l'entrée de type `foaf:Person` aimerait :

```
?x a foaf:Person .
?x   <http://www.example.org/likes> ?m .
?m a <http://www.example.org/movie> .

## Query
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_PREDICTION' ;
                      neptune-ml:input ?input ;
```

```
                        neptune-ml:predicate <http://www.example.org/likes> ;
                        neptune-ml:output ?output ;
                        neptune-ml:outputClass <http://www.example.org/movie> .
  }
}
```

La requête elle-même peut être personnalisée comme suit :

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-user-movie-prediction-
endpoint' ;
                      neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                      neptune-ml:limit "5"^^xsd:integer ;
                      neptune-ml:batchSize "40"^^xsd:integer ;
                      neptune-ml:threshold "0.1"^^xsd:double ;
                      neptune-ml:timeout "1000"^^xsd:integer ;
                      neptune-ml:outputScore ?score ;

                      neptune-ml:modelType 'OBJECT_PREDICTION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/likes> ;
                      neptune-ml:output ?output ;
                      neptune-ml:outputClass <http://www.example.org/movie> .
  }
}
```

Exemple de prédiction de sujet SPARQL

La prédiction de sujet prédit le sujet pour un prédicat et un objet donnés.

Par exemple, la requête suivante prédit qui (de type `foaf:User`) regardera un film donné :

```
SELECT * WHERE { ?input (a foaf:Movie) .
    SERVICE neptune-ml:inference {
        neptune-ml:config neptune-ml:modelType 'SUBJECT_PREDICTION' ;
                          neptune-ml:input ?input ;
                          neptune-ml:predicate <http://aws.amazon.com/neptune/csv2rdf/
object_Property/rated> ;
                          neptune-ml:output ?output ;
```

```
                        neptune-ml:outputClass <http://aws.amazon.com/neptune/
csv2rdf/class/User> ;       }
}
```

Liste des exceptions pour les requêtes d'inférence SPARQL de Neptune ML

- **BadRequestException** – Message : The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects at least 1 value for the parameter *(parameter name)*, found zero.
- **BadRequestException** – Message : The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects at most 1 value for the parameter *(parameter name)*, found *(a number)* values.
- **BadRequestException** – Message : Invalid predicate *(predicate name)* provided for external service http://aws.amazon.com/neptune/vocab/v01/services/ml#inference query.
- **BadRequestException** – Message : The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the predicate *(predicate name)* to be defined.
- **BadRequestException** – Message : The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the value of (parameter) *(parameter name)* to be a variable, found: *(type)*"
- **BadRequestException** – Message : The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the input *(parameter name)* to be a constant, found: *(type)*.
- **BadRequestException** – Message : The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference is expected to return only 1 value.
- **BadRequestException** – Message : "The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference only allows StatementPatternNodes.
- **BadRequestException** – Message : The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference does not allow the predicate *(predicate name)*.
- **BadRequestException** – Message : The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference predicates cannot be variables, found: *(type)*.

- **BadRequestException** – Message : The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference predicates are expected to be part of the namespace *(namespace name)*, found: *(namespace name)*.

Référence de l'API de gestion Neptune ML

Table des matières

- Traitement des données à l'aide de la commande dataprocessing
  - Création d'une tâche de traitement de données à l'aide de la commande Neptune ML dataprocessing
  - Obtention du statut d'une tâche de traitement de données à l'aide de la commande Neptune ML dataprocessing
  - Arrêt d'une tâche de traitement de données à l'aide de la commande Neptune ML dataprocessing
  - Répertorier les tâches de traitement de données actives à l'aide de la commande Neptune ML dataprocessing
- Entraînement de modèle à l'aide de la commande modeltraining
  - Création d'une tâche d'entraînement de modèle à l'aide de la commande Neptune ML modeltraining
  - Obtention du statut d'une tâche d'entraînement de modèle à l'aide de la commande Neptune ML modeltraining
  - Arrêt d'une tâche d'entraînement de modèle à l'aide de la commande Neptune ML modeltraining
  - Répertorier les tâches d'entraînement de modèle actives à l'aide de la commande Neptune ML modeltraining
- Transformation de modèle à l'aide de la commande modeltransform
  - Création d'une tâche de transformation de modèle à l'aide de la commande Neptune ML modeltransform
  - Obtention du statut d'une tâche de transformation de modèle à l'aide de la commande Neptune ML modeltransform
  - Arrêt d'une tâche de transformation de modèle à l'aide de la commande Neptune ML modeltransform
  - Répertorier les tâches de transformation de modèle actives à l'aide de la commande Neptune ML modeltransform
- Gestion des points de terminaison d'inférence à l'aide de la commande endpoints
  - Création d'un point de terminaison d'inférence à l'aide de la commande Neptune ML endpoints
  - Obtention du statut d'un point de terminaison d'inférence à l'aide de la commande Neptune ML endpoints

  - Suppression d'un point de terminaison d'instance à l'aide de la commande Neptune ML endpoints
  - Répertorier les points de terminaison d'inférence à l'aide de la commande Neptune ML endpoints
- Erreurs et exceptions de l'API de gestion Neptune ML

Traitement des données à l'aide de la commande **dataprocessing**

Vous utilisez la commande Neptune ML `dataprocessing` pour créer une tâche de traitement de données, vérifier son statut, l'arrêter ou répertorier toutes les tâches de traitement de données actives.

Création d'une tâche de traitement de données à l'aide de la commande Neptune ML **dataprocessing**

Une commande Neptune ML `dataprocessing` typique pour créer une nouvelle tâche ressemble à ceci :

AWS CLI

```
aws neptunedata start-ml-data-processing-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --input-data-s3-location "s3://(S3 bucket name)/(path to your input folder)" \
  --id "(a job ID for the new job)" \
  --processed-data-s3-location "s3://(S3 bucket name)/(path to your output folder)"
```

Pour plus d'informations, consultez start-ml-data-processing-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_data_processing_job(
    inputDataS3Location='s3://(S3 bucket name)/(path to your input folder)',
    id='(a job ID for the new job)',
    processedDataS3Location='s3://(S3 bucket name)/(path to your output folder)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/dataprocessing \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(S3 bucket name)/(path to your input folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(S3 bucket name)/(path to your input folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)"
      }'
```

Une commande pour lancer un retraitement incrémentiel ressemble à ceci :

AWS CLI

```
aws neptunedata start-ml-data-processing-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --input-data-s3-location "s3://(S3 bucket name)/(path to your input folder)" \
  --id "(a job ID for this job)" \
```

```
  --processed-data-s3-location "s3://(S3 bucket name)/(path to your output folder)"
 \
  --previous-data-processing-job-id "(the job ID of a previously completed job to
 update)"
```

Pour plus d'informations, consultez start-ml-data-processing-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_data_processing_job(
    inputDataS3Location='s3://(S3 bucket name)/(path to your input folder)',
    id='(a job ID for this job)',
    processedDataS3Location='s3://(S3 bucket name)/(path to your output folder)',
    previousDataProcessingJobId='(the job ID of a previously completed job to
 update)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/dataprocessing \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(S3 bucket name)/(path to your input folder)",
        "id" : "(a job ID for this job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)",
        "previousDataProcessingJobId" : "(the job ID of a previously completed job
 to update)"
```

```
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(S3 bucket name)/(path to your input folder)",
        "id" : "(a job ID for this job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)",
        "previousDataProcessingJobId" : "(the job ID of a previously completed job
 to update)"
      }'
```

Paramètres de création d'une tâche **`dataprocessing`**

- **`id`** : (facultatif) identifiant unique de la nouvelle tâche.

  Type : string. Valeur par défaut : un UUID généré automatiquement.
- **`previousDataProcessingJobId`** : (facultatif) ID de tâche d'une tâche de traitement de données terminée, exécutée sur une version antérieure des données.

  Type : string. Valeur par défaut : aucune.

  Remarque : Utilisez-le pour le traitement incrémentiel des données, afin de mettre à jour le modèle quand les données du graphe ont changé (mais pas quand les données ont été supprimées).
- **`inputDataS3Location`**— (Obligatoire) L'URI de l'emplacement Amazon S3 où vous souhaitez qu' SageMaker AI télécharge les données nécessaires à l'exécution de la tâche de traitement des données.

  Type : string.

- **`processedDataS3Location`**— (Obligatoire) L'URI de l'emplacement Amazon S3 où vous souhaitez qu' SageMaker AI enregistre les résultats d'une tâche de traitement de données.

  Type : string.

- **`sagemakerIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM pour l'exécution de l' SageMaker IA.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.

- **`neptuneIamRoleArn`**— (Facultatif) Le nom de ressource Amazon (ARN) d'un rôle IAM que l' SageMaker IA peut assumer pour effectuer des tâches en votre nom.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.

- **`processingInstanceType`** : (facultatif) type d'instance ML utilisé lors du traitement des données. Sa mémoire doit être suffisamment grande pour contenir le jeu de données traité.

  Type : string. Valeur par défaut : le plus petit type d'instance `ml.r5` dont la mémoire est dix fois supérieure à la taille des données de graphe exportées sur le disque.

  Remarque : Neptune ML peut sélectionner le type d'instance automatiquement. Consultez Sélection d'une instance pour le traitement des données.

- **`processingInstanceVolumeSizeInGB`** : (facultatif) taille du volume de disque de l'instance de traitement. Les données d'entrée et les données traitées étant toutes stockées sur disque, la taille du volume doit être suffisante pour contenir les deux jeux de données.

  Type : entier. Par défaut : `0`.

  Remarque : Si ce paramètre n'est pas spécifié ou s'il est égal à 0, Neptune ML choisit automatiquement la taille du volume en fonction de la taille des données.

- **`processingTimeOutInSeconds`** : (facultatif) délai d'expiration en secondes pour la tâche de traitement des données.

  Type : entier. Valeur par défaut : `86,400` (1 jour).

- **`modelType`** : (facultatif) l'un des deux types de modèles actuellement pris en charge par Neptune ML : modèles de graphes hétérogènes (`heterogeneous`) et graphe de connaissances (`kge`).

  Type : string. Valeur par défaut : aucune.

Remarque : Si ce paramètre n'est pas spécifié, Neptune ML choisit automatiquement le type de modèle en fonction des données.

- **`configFileName`** : (facultatif) fichier de spécification de données qui décrit comment charger les données de graphe exportées à des fins d'entraînement. Ce fichier est automatiquement généré par la boîte à outils d'exportation Neptune.

  Type : string. Par défaut : `training-data-configuration.json`.
- **`subnets`**— (Facultatif) Les IDs sous-réseaux du VPC Neptune.

  Type : liste de chaînes. Valeur par défaut : aucune.
- **`securityGroupIds`**— (Facultatif) Le groupe de sécurité VPC. IDs

  Type : liste de chaînes. Valeur par défaut : aucune.
- **`volumeEncryptionKMSKey`**— (Facultatif) La clé AWS Key Management Service (AWS KMS) utilisée par SageMaker AI pour chiffrer les données sur le volume de stockage attaché aux instances de calcul ML qui exécutent la tâche de traitement.

  Type : string. Valeur par défaut : aucune.
- **`enableInterContainerTrafficEncryption`** : (facultatif) activez ou désactivez le chiffrement du trafic entre conteneurs dans les tâches d'entraînement ou de réglage des hyperparamètres.

  Type : booléen. Valeur par défaut : True.

  > **Note**
  >
  > Le paramètre `enableInterContainerTrafficEncryption` est disponible uniquement dans la version 1.2.0.2.R3 du moteur.
- **`s3OutputEncryptionKMSKey`**— (Facultatif) La clé AWS Key Management Service (AWS KMS) utilisée par l' SageMaker IA pour chiffrer le résultat du travail de formation.

  Type : string. Valeur par défaut : aucune.

Obtention du statut d'une tâche de traitement de données à l'aide de la commande Neptune ML **`dataprocessing`**

Voici un exemple de commande Neptune ML `dataprocessing` pour obtenir le statut d'une tâche :

AWS CLI

```
aws neptunedata get-ml-data-processing-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)"
```

Pour plus d'informations, consultez get-ml-data-processing-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.get_ml_data_processing_job(
    id='(the job ID)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/dataprocessing/(the job ID) \
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -s \
```

```
  "https://your-neptune-endpoint:port/ml/dataprocessing/(the job ID)" \
  | python -m json.tool
```

Paramètres d'obtention du statut de la tâche **`dataprocessing`**

- **`id`** : (obligatoire) identifiant unique de la tâche de traitement de données.

  Type : string.
- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM qui permet à Neptune d'accéder SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.

Arrêt d'une tâche de traitement de données à l'aide de la commande Neptune ML **`dataprocessing`**

Voici un exemple de commande Neptune ML `dataprocessing` permettant d'arrêter une tâche :

AWS CLI

```
aws neptunedata cancel-ml-data-processing-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)"
```

Pour également nettoyer les artefacts Amazon S3, procédez comme suit :

```
aws neptunedata cancel-ml-data-processing-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)" \
  --clean
```

Pour plus d'informations, consultez cancel-ml-data-processing-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config
```

```
client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.cancel_ml_data_processing_job(
    id='(the job ID)',
    clean=True
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/dataprocessing/(the job ID) \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

Pour également nettoyer les artefacts Amazon S3, procédez comme suit :

```
awscurl "https://your-neptune-endpoint:port/ml/dataprocessing/(the job ID)?
clean=true" \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/dataprocessing/(the job ID)"
```

Ou encore :

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/dataprocessing/(the job ID)?
clean=true"
```

Paramètres d'arrêt de la tâche **`dataprocessing`**

- **`id`** : (obligatoire) identifiant unique de la tâche de traitement de données.

  Type : string.
- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM qui permet à Neptune d'accéder SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.
- **`clean`** : (facultatif) cet indicateur spécifie que tous les artefacts Amazon S3 doivent être supprimés lorsque la tâche est arrêtée.

  Type : booléen. Par défaut : `FALSE`.

Répertorier les tâches de traitement de données actives à l'aide de la commande Neptune ML **`dataprocessing`**

Voici un exemple de commande Neptune ML `dataprocessing` permettant de répertorier les tâches actives :

AWS CLI

```
aws neptunedata list-ml-data-processing-jobs \
  --endpoint-url https://your-neptune-endpoint:port
```

Pour limiter le nombre de résultats :

```
aws neptunedata list-ml-data-processing-jobs \
  --endpoint-url https://your-neptune-endpoint:port \
  --max-items 3
```

Pour plus d'informations, consultez list-ml-data-processing-jobs dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.list_ml_data_processing_jobs(
    maxItems=3
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/dataprocessing \
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

Pour limiter le nombre de résultats :

```
awscurl "https://your-neptune-endpoint:port/ml/dataprocessing?maxItems=3" \
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -s "https://your-neptune-endpoint:port/ml/dataprocessing"
```

Ou encore :

```
curl -s "https://your-neptune-endpoint:port/ml/dataprocessing?maxItems=3"
```

Paramètres pour répertorier les tâches **`dataprocessing`**

- **`maxItems`** : (facultatif) nombre maximal d'éléments à renvoyer.

  Type : entier. Par défaut : `10`. Valeur maximale autorisée : `1024`.
- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM qui permet à Neptune d'accéder SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.

Entraînement de modèle à l'aide de la commande **`modeltraining`**

Vous utilisez la commande Neptune ML `modeltraining` pour créer une tâche d'entraînement de modèle, vérifier son statut, l'arrêter ou répertorier toutes les tâches d'entraînement de modèle actives.

Création d'une tâche d'entraînement de modèle à l'aide de la commande Neptune ML **`modeltraining`**

Une commande Neptune ML `modeltraining` permettant de créer une toute nouvelle tâche ressemble à ceci :

AWS CLI

```
aws neptunedata start-ml-model-training-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique model-training job ID)" \
  --data-processing-job-id "(the data-processing job-id of a completed job)" \
  --train-model-s3-location "s3://(your S3 bucket)/neptune-model-graph-autotrainer"
```

Pour plus d'informations, consultez start-ml-model-training-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_model_training_job(
    id='(a unique model-training job ID)',
    dataProcessingJobId='(the data-processing job-id of a completed job)',
    trainModelS3Location='s3://(your S3 bucket)/neptune-model-graph-autotrainer'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltraining \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your S3 bucket)/neptune-model-graph-
autotrainer"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/modeltraining \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your S3 bucket)/neptune-model-graph-
autotrainer"
      }'
```

Une commande Neptune ML `modeltraining` permettant de créer une tâche de mise à jour pour un entraînement de modèle incrémentiel ressemble à ceci :

AWS CLI

```
aws neptunedata start-ml-model-training-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique model-training job ID)" \
```

```
  --data-processing-job-id "(the data-processing job-id of a completed job)" \
  --train-model-s3-location "s3://(your S3 bucket)/neptune-model-graph-autotrainer"
 \
  --previous-model-training-job-id "(the job ID of a completed model-training job to
 update)"
```

Pour plus d'informations, consultez start-ml-model-training-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_model_training_job(
    id='(a unique model-training job ID)',
    dataProcessingJobId='(the data-processing job-id of a completed job)',
    trainModelS3Location='s3://(your S3 bucket)/neptune-model-graph-autotrainer',
    previousModelTrainingJobId='(the job ID of a completed model-training job to
 update)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltraining \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your S3 bucket)/neptune-model-graph-
autotrainer",
```

```
        "previousModelTrainingJobId" : "(the job ID of a completed model-training
 job to update)"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/modeltraining \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your S3 bucket)/neptune-model-graph-
autotrainer",
        "previousModelTrainingJobId" : "(the job ID of a completed model-training
 job to update)"
      }'
```

Une commande Neptune ML `modeltraining` permettant de créer une nouvelle tâche avec l'implémentation d'un modèle personnalisé fourni par l'utilisateur ressemble à ceci :

AWS CLI

```
aws neptunedata start-ml-model-training-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique model-training job ID)" \
  --data-processing-job-id "(the data-processing job-id of a completed job)" \
  --train-model-s3-location "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer" \
  --model-name "custom" \
  --custom-model-training-parameters '{
    "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
    "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
```

```
    "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
  }'
```

Pour plus d'informations, consultez start-ml-model-training-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_model_training_job(
    id='(a unique model-training job ID)',
    dataProcessingJobId='(the data-processing job-id of a completed job)',
    trainModelS3Location='s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer',
    modelName='custom',
    customModelTrainingParameters={
        'sourceS3DirectoryPath': 's3://(your Amazon S3 bucket)/(path to your Python
 module)',
        'trainingEntryPointScript': '(your training script entry-point name in the
 Python module)',
        'transformEntryPointScript': '(your transform script entry-point name in the
 Python module)'
    }
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltraining \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
```

```
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer",
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your
 Python module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in
 the Python module)"
        }
     }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/modeltraining \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer",
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your
 Python module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in
 the Python module)"
        }
     }'
```

Paramètres de création d'une tâche **`modeltraining`**

- **`id`** : (facultatif) identifiant unique de la nouvelle tâche.

  Type : string. Valeur par défaut : un UUID généré automatiquement.
- **`dataProcessingJobId`** : (obligatoire) ID de la tâche de traitement de données terminée qui a créé les données qui seront utilisées par l'entraînement.

  Type : string.
- **`trainModelS3Location`** : (obligatoire) emplacement dans Amazon S3 où les artefacts de modèle doivent être stockés.

  Type : string.
- **`previousModelTrainingJobId`** : (facultatif) ID de tâche d'une tâche d'entraînement de modèle terminée que vous souhaitez actualiser progressivement en fonction des données mises à jour.

  Type : string. Valeur par défaut : aucune.
- **`sagemakerIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM pour l'exécution de l' SageMaker IA.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.
- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM qui permet à Neptune d'accéder SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.
- **`modelName`** : (facultatif) type de modèle pour l'entraînement. Par défaut, le modèle ML est automatiquement basé sur le `modelType` utilisé dans le cadre du traitement de données, mais vous pouvez spécifier ici un autre type de modèle.

  Type : string. Valeur par défaut : `rgcn` pour les graphes hétérogènes et `kge` pour les graphes de connaissances. Valeurs valides : pour les graphes hétérogènes : `rgcn`. Pour les graphes `kge` : `transe`, `distmult` ou `rotate`. Pour l'implémentation d'un modèle personnalisé : `custom`.
- **`baseProcessingInstanceType`** : (facultatif) type d'instance ML utilisé pour préparer et gérer l'entraînement de modèles ML.

Type : string. Remarque : Il s'agit d'une instance de CPU choisie en fonction des besoins en mémoire pour le traitement des données et du modèle d'entraînement. Consultez Sélection d'une instance pour l'entraînement de modèle et la transformation de modèle.

- **`trainingInstanceType`** : (facultatif) type d'instance ML utilisé pour l'entraînement de modèle. Tous les modèles Neptune ML prennent en charge l'entraînement de CPU, de GPU et de multiGPU.

  Type : string. Par défaut : `ml.p3.2xlarge`.

  Remarque : Le choix du type d'instance approprié pour l'entraînement dépend du type de tâche, de la taille de graphe et de votre budget. Consultez Sélection d'une instance pour l'entraînement de modèle et la transformation de modèle.

- **`trainingInstanceVolumeSizeInGB`** : (facultatif) taille du volume de disque de l'instance d'entraînement. Les données d'entrée et le modèle de sortie étant toutes stockées sur disque, la taille du volume doit être suffisante pour contenir les deux jeux de données.

  Type : entier. Par défaut : `0`.

  Remarque : Si elle n'est pas spécifiée ou si elle est égale à 0, Neptune ML sélectionne une taille de volume de disque en fonction de la recommandation générée lors de l'étape de traitement de données. Consultez Sélection d'une instance pour l'entraînement de modèle et la transformation de modèle.

- **`trainingTimeOutInSeconds`** : (facultatif) délai d'attente en secondes de la tâche d'entraînement.

  Type : entier. Valeur par défaut : `86,400` (1 jour).

- **`maxHPONumberOfTrainingJobs`** – Nombre total maximal de tâches d'entraînement à démarrer pour la tâche de réglage des hyperparamètres.

  Type : entier. Par défaut : `2`.

  Remarque : Neptune ML règle automatiquement les hyperparamètres du modèle de machine learning. Pour obtenir un modèle performant, utilisez au moins 10 tâches (en d'autres termes, définissez `maxHPONumberOfTrainingJobs` sur 10). En général, plus le réglage est long, meilleurs sont les résultats.

- **`maxHPOParallelTrainingJobs`** – Nombre maximal de tâches d'entraînement parallèles à démarrer pour la tâche de réglage des hyperparamètres.

Type : entier. Par défaut : 2.

Remarque : Le nombre de tâches parallèles que vous pouvez exécuter est limité par les ressources disponibles sur votre instance d'entraînement.

- **`subnets`**— (Facultatif) Les IDs sous-réseaux du VPC Neptune.

  Type : liste de chaînes. Valeur par défaut : aucune.

- **`securityGroupIds`**— (Facultatif) Le groupe de sécurité VPC. IDs

  Type : liste de chaînes. Valeur par défaut : aucune.

- **`volumeEncryptionKMSKey`**— (Facultatif) La clé AWS Key Management Service (AWS KMS) utilisée par l' SageMaker IA pour chiffrer les données sur le volume de stockage attaché aux instances de calcul ML qui exécutent la tâche de formation.

  Type : string. Valeur par défaut : aucune.

- **`s3OutputEncryptionKMSKey`**— (Facultatif) La clé AWS Key Management Service (AWS KMS) utilisée par l' SageMaker IA pour chiffrer le résultat de la tâche de traitement.

  Type : string. Valeur par défaut : aucune.

- **`enableInterContainerTrafficEncryption`** : (facultatif) activez ou désactivez le chiffrement du trafic entre conteneurs dans les tâches d'entraînement ou de réglage des hyperparamètres.

  Type : booléen. Valeur par défaut : True.

  > **Note**
  >
  > Le paramètre `enableInterContainerTrafficEncryption` est disponible uniquement dans la version 1.2.0.2.R3 du moteur.

- **`enableManagedSpotTraining`** : (facultatif) optimise le coût d'entraînement de modèles de machine learning à l'aide d'instances Spot Amazon Elastic Compute Cloud.. Pour plus d'informations, consultez Managed Spot Training sur Amazon SageMaker.

  Type : booléen. Valeur par défaut : False.

- **`customModelTrainingParameters`** : (facultatif) configuration pour un entraînement de modèle personnalisé. Ceci est un objet JSON avec les champs suivants :

- **`sourceS3DirectoryPath`** : (obligatoire) chemin de l'emplacement Amazon S3 où se trouve le module Python implémentant votre modèle. Il doit pointer vers un emplacement Amazon S3 existant valide contenant, au minimum, un script d'entraînement, un script de transformation et un fichier `model-hpo-configuration.json`.
- **`trainingEntryPointScript`** : (facultatif) nom du point d'entrée dans votre module d'un script qui effectue l'entraînement de modèle et utilise les hyperparamètres comme arguments de ligne de commande, y compris les hyperparamètres fixes.

  Par défaut : `training.py`.
- **`transformEntryPointScript`** : (facultatif) nom du point d'entrée dans le module d'un script qui doit être exécuté une fois que le modèle le plus approprié issu de la recherche par hyperparamètres a été identifié, afin de calculer les artefacts de modèle nécessaires au déploiement du modèle. Il devrait pouvoir s'exécuter sans arguments de ligne de commande.

  Par défaut : `transform.py`.
- **`maxWaitTime`** : (facultatif) temps d'attente maximal, en secondes, lors de l'entraînement de modèle à l'aide d'instances Spot. Devrait être supérieur à `trainingTimeOutInSeconds`.

  Type : entier.

Obtention du statut d'une tâche d'entraînement de modèle à l'aide de la commande Neptune ML **`modeltraining`**

Voici un exemple de commande Neptune ML `modeltraining` pour obtenir le statut d'une tâche :

AWS CLI

```
aws neptunedata get-ml-model-training-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)"
```

Pour plus d'informations, consultez get-ml-model-training-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config
```

```
client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.get_ml_model_training_job(
    id='(the job ID)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltraining/(the job ID) \
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -s \
  "https://your-neptune-endpoint:port/ml/modeltraining/(the job ID)" \
  | python -m json.tool
```

Paramètres d'obtention du statut de la tâche **`modeltraining`**

- **`id`** : (obligatoire) identifiant unique de la tâche d'entraînement de modèle.

  Type : string.

- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM qui permet à Neptune d'accéder SageMaker aux ressources AI et Amazon S3.

Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.

Arrêt d'une tâche d'entraînement de modèle à l'aide de la commande Neptune ML `modeltraining`

Voici un exemple de commande Neptune ML `modeltraining` permettant d'arrêter une tâche :

AWS CLI

```
aws neptunedata cancel-ml-model-training-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)"
```

Pour également nettoyer les artefacts Amazon S3 :

```
aws neptunedata cancel-ml-model-training-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)" \
  --clean
```

Pour plus d'informations, consultez cancel-ml-model-training-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.cancel_ml_model_training_job(
    id='(the job ID)',
    clean=True
)
```

```
print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltraining/(the job ID) \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

Pour également nettoyer les artefacts Amazon S3 :

```
awscurl "https://your-neptune-endpoint:port/ml/modeltraining/(the job ID)?
clean=true" \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/modeltraining/(the job ID)"
```

Ou encore :

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/modeltraining/(the job ID)?
clean=true"
```

Paramètres d'arrêt de la tâche **`modeltraining`**

- **`id`** : (obligatoire) identifiant unique de la tâche d'entraînement de modèle.

  Type : string.

- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM qui permet à Neptune d'accéder SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.
- **`clean`** : (facultatif) cet indicateur spécifie que tous les artefacts Amazon S3 doivent être supprimés lorsque la tâche est arrêtée.

  Type : booléen. Par défaut : `FALSE`.

Répertorier les tâches d'entraînement de modèle actives à l'aide de la commande Neptune ML **`modeltraining`**

Voici un exemple de commande Neptune ML `modeltraining` permettant de répertorier les tâches actives :

AWS CLI

```
aws neptunedata list-ml-model-training-jobs \
  --endpoint-url https://your-neptune-endpoint:port
```

Pour limiter le nombre de résultats :

```
aws neptunedata list-ml-model-training-jobs \
  --endpoint-url https://your-neptune-endpoint:port \
  --max-items 3
```

Pour plus d'informations, consultez list-ml-model-training-jobs dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)
```

```
response = client.list_ml_model_training_jobs(
    maxItems=3
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltraining \
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

Pour limiter le nombre de résultats :

```
awscurl "https://your-neptune-endpoint:port/ml/modeltraining?maxItems=3" \
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -s "https://your-neptune-endpoint:port/ml/modeltraining" | python -m json.tool
```

Ou encore :

```
curl -s "https://your-neptune-endpoint:port/ml/modeltraining?maxItems=3" | python -m
 json.tool
```

Paramètres pour répertorier les tâches **`modeltraining`**

- **`maxItems`** : (facultatif) nombre maximal d'éléments à renvoyer.

Type : entier. Par défaut : `10`. Valeur maximale autorisée : `1024`.

- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM qui permet à Neptune d'accéder SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.

Transformation de modèle à l'aide de la commande **`modeltransform`**

Vous utilisez la commande Neptune ML `modeltransform` pour créer une tâche de transformation de modèle, vérifier son statut, l'arrêter ou répertorier toutes les tâches de transformation de modèle actives.

Création d'une tâche de transformation de modèle à l'aide de la commande Neptune ML **`modeltransform`**

Une commande Neptune ML `modeltransform` permettant de créer une tâche de transformation incrémentielle, sans réentraînement de modèle, ressemble à ceci :

AWS CLI

```
aws neptunedata start-ml-model-transform-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique model-transform job ID)" \
  --data-processing-job-id "(the job-id of a completed data-processing job)" \
  --ml-model-training-job-id "(the job-id of a completed model-training job)" \
  --model-transform-output-s3-location "s3://(your S3 bucket)/neptune-model-
transform"
```

Pour plus d'informations, consultez start-ml-model-transform-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_model_transform_job(
    id='(a unique model-transform job ID)',
    dataProcessingJobId='(the job-id of a completed data-processing job)',
    mlModelTrainingJobId='(the job-id of a completed model-training job)',
    modelTransformOutputS3Location='s3://(your S3 bucket)/neptune-model-transform'
```

```
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltransform \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "dataProcessingJobId" : "(the job-id of a completed data-processing job)",
        "mlModelTrainingJobId" : "(the job-id of a completed model-training job)",
        "modelTransformOutputS3Location" : "s3://(your S3 bucket)/neptune-model-
transform"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/modeltransform \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "dataProcessingJobId" : "(the job-id of a completed data-processing job)",
        "mlModelTrainingJobId" : "(the job-id of a completed model-training job)",
        "modelTransformOutputS3Location" : "s3://(your S3 bucket)/neptune-model-
transform"
      }'
```

Voici une `modeltransform` commande Neptune ML permettant de créer une tâche à partir d'une tâche de formation à l' SageMaker IA terminée :

AWS CLI

```
aws neptunedata start-ml-model-transform-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique model-transform job ID)" \
  --training-job-name "(name of a completed SageMaker training job)" \
  --model-transform-output-s3-location "s3://(your S3 bucket)/neptune-model-
transform"
```

Pour plus d'informations, consultez start-ml-model-transform-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_model_transform_job(
    id='(a unique model-transform job ID)',
    trainingJobName='(name of a completed SageMaker training job)',
    modelTransformOutputS3Location='s3://(your S3 bucket)/neptune-model-transform'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltransform \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your S3 bucket)/neptune-model-
transform"
```

```
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/modeltransform \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your S3 bucket)/neptune-model-
transform"
      }'
```

Une commande Neptune ML `modeltransform` permettant de créer une tâche qui utilise une implémentation de modèle personnalisée ressemble à ceci :

AWS CLI

```
aws neptunedata start-ml-model-transform-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique model-transform job ID)" \
  --training-job-name "(name of a completed SageMaker training job)" \
  --model-transform-output-s3-location "s3://(your Amazon S3 bucket)/neptune-model-
transform/" \
  --custom-model-transform-parameters '{
    "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
    "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
  }'
```

Pour plus d'informations, consultez start-ml-model-transform-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.start_ml_model_transform_job(
    id='(a unique model-transform job ID)',
    trainingJobName='(name of a completed SageMaker training job)',
    modelTransformOutputS3Location='s3://(your Amazon S3 bucket)/neptune-model-
transform/',
    customModelTransformParameters={
        'sourceS3DirectoryPath': 's3://(your Amazon S3 bucket)/(path to your Python
 module)',
        'transformEntryPointScript': '(your transform script entry-point name in the
 Python module)'
    }
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltransform \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-
model-transform/",
        "customModelTransformParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in
 the Python module)"
```

```
        }
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/modeltransform \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-
model-transform/",
        "customModelTransformParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in
 the Python module)"
        }
      }'
```

Paramètres de création d'une tâche **`modeltransform`**

- **`id`** : (facultatif) identifiant unique de la nouvelle tâche.

  Type : string. Valeur par défaut : un UUID généré automatiquement.

- **`dataProcessingJobId`** – ID de tâche d'une tâche de traitement de données terminée.

  Type : string.

  Remarque : Vous devez inclure `dataProcessingJobId` et `mlModelTrainingJobId`, ou `trainingJobName`.

- **`mlModelTrainingJobId`** – ID de tâche d'une tâche d'entraînement de modèle terminée.

Type : string.

Remarque : Vous devez inclure `dataProcessingJobId` et `mlModelTrainingJobId`, ou `trainingJobName`.

- **`trainingJobName`**— Le nom d'un poste de formation en SageMaker IA terminé.

  Type : string.

  Remarque : Vous devez inclure à la fois les paramètres `dataProcessingJobId` et `mlModelTrainingJobId`, ou le paramètre `trainingJobName`.

- **`sagemakerIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM pour l'exécution de l' SageMaker IA.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.

- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM qui permet à Neptune d'accéder SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.

- **`customModelTransformParameters`** : (facultatif) informations de configuration d'une transformation de modèle à l'aide d'un modèle personnalisé. L'objet `customModelTransformParameters` contient les champs suivants, dont les valeurs doivent être compatibles avec les paramètres de modèle enregistrés lors de la tâche d'entraînement :

  - **`sourceS3DirectoryPath`** : (obligatoire) chemin de l'emplacement Amazon S3 où se trouve le module Python implémentant votre modèle. Il doit pointer vers un emplacement Amazon S3 existant valide contenant, au minimum, un script d'entraînement, un script de transformation et un fichier `model-hpo-configuration.json`.
  - **`transformEntryPointScript`** : (facultatif) nom du point d'entrée dans le module d'un script qui doit être exécuté une fois que le modèle le plus approprié issu de la recherche par hyperparamètres a été identifié, afin de calculer les artefacts de modèle nécessaires au déploiement du modèle. Il devrait pouvoir s'exécuter sans arguments de ligne de commande.

    Par défaut : `transform.py`.

- **`baseProcessingInstanceType`** : (facultatif) type d'instance ML utilisé pour préparer et gérer l'entraînement de modèles ML.

Type : string. Remarque : Il s'agit d'une instance de CPU choisie en fonction des besoins en mémoire pour le traitement des données et du modèle de transformation. Consultez Sélection d'une instance pour l'entraînement de modèle et la transformation de modèle.

- **`baseProcessingInstanceVolumeSizeInGB`** : (facultatif) taille du volume de disque de l'instance d'entraînement. Les données d'entrée et le modèle de sortie étant toutes stockées sur disque, la taille du volume doit être suffisante pour contenir les deux jeux de données.

  Type : entier. Par défaut : `0`.

  Remarque : Si elle n'est pas spécifiée ou si elle est égale à 0, Neptune ML sélectionne une taille de volume de disque en fonction de la recommandation générée lors de l'étape de traitement de données. Consultez Sélection d'une instance pour l'entraînement de modèle et la transformation de modèle.

- **`subnets`**— (Facultatif) Les IDs sous-réseaux du VPC Neptune.

  Type : liste de chaînes. Valeur par défaut : aucune.

- **`securityGroupIds`**— (Facultatif) Le groupe de sécurité VPC. IDs

  Type : liste de chaînes. Valeur par défaut : aucune.

- **`volumeEncryptionKMSKey`**— (Facultatif) La clé AWS Key Management Service (AWS KMS) utilisée par SageMaker AI pour chiffrer les données sur le volume de stockage attaché aux instances de calcul ML qui exécutent la tâche de transformation.

  Type : string. Valeur par défaut : aucune.

- **`enableInterContainerTrafficEncryption`** : (facultatif) activez ou désactivez le chiffrement du trafic entre conteneurs dans les tâches d'entraînement ou de réglage des hyperparamètres.

  Type : booléen. Valeur par défaut : True.

  > **Note**
  >
  > Le paramètre `enableInterContainerTrafficEncryption` est disponible uniquement dans la version 1.2.0.2.R3 du moteur.

- **`s3OutputEncryptionKMSKey`**— (Facultatif) La clé AWS Key Management Service (AWS KMS) utilisée par l' SageMaker IA pour chiffrer le résultat de la tâche de traitement.

  Type : string. Valeur par défaut : aucune.

Obtention du statut d'une tâche de transformation de modèle à l'aide de la commande Neptune ML **`modeltransform`**

Voici un exemple de commande Neptune ML `modeltransform` pour obtenir le statut d'une tâche :

AWS CLI

```
aws neptunedata get-ml-model-transform-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)"
```

Pour plus d'informations, consultez get-ml-model-transform-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.get_ml_model_transform_job(
    id='(the job ID)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltransform/(the job ID) \
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -s \
  "https://your-neptune-endpoint:port/ml/modeltransform/(the job ID)" \
  | python -m json.tool
```

Paramètres d'obtention du statut de la tâche **`modeltransform`**

- **`id`** : (obligatoire) identifiant unique de la tâche de transformation de modèle.

  Type : string.
- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM qui permet à Neptune d'accéder SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.

Arrêt d'une tâche de transformation de modèle à l'aide de la commande Neptune ML **`modeltransform`**

Voici un exemple de commande Neptune ML `modeltransform` permettant d'arrêter une tâche :

AWS CLI

```
aws neptunedata cancel-ml-model-transform-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)"
```

Pour également nettoyer les artefacts Amazon S3 :

```
aws neptunedata cancel-ml-model-transform-job \
  --endpoint-url https://your-neptune-endpoint:port \
```

```
  --id "(the job ID)" \
  --clean
```

Pour plus d'informations, consultez cancel-ml-model-transform-job dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.cancel_ml_model_transform_job(
    id='(the job ID)',
    clean=True
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltransform/(the job ID) \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

Pour également nettoyer les artefacts Amazon S3 :

```
awscurl "https://your-neptune-endpoint:port/ml/modeltransform/(the job ID)?
clean=true" \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/modeltransform/(the job ID)"
```

Ou encore :

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/modeltransform/(the job ID)?
clean=true"
```

Paramètres d'arrêt de la tâche **`modeltransform`**

- **`id`** : (obligatoire) identifiant unique de la tâche de transformation de modèle.

  Type : string.
- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM qui permet à Neptune d'accéder SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.
- **`clean`** : (facultatif) cet indicateur spécifie que tous les artefacts Amazon S3 doivent être supprimés lorsque la tâche est arrêtée.

  Type : booléen. Par défaut : `FALSE`.

Répertorier les tâches de transformation de modèle actives à l'aide de la commande Neptune ML **`modeltransform`**

Voici un exemple de commande Neptune ML `modeltransform` permettant de répertorier les tâches actives :

AWS CLI

```
aws neptunedata list-ml-model-transform-jobs \
  --endpoint-url https://your-neptune-endpoint:port
```

Pour limiter le nombre de résultats :

```
aws neptunedata list-ml-model-transform-jobs \
  --endpoint-url https://your-neptune-endpoint:port \
  --max-items 3
```

Pour plus d'informations, consultez list-ml-model-transform-jobs dans le manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.list_ml_model_transform_jobs(
    maxItems=3
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/modeltransform \
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

Pour limiter le nombre de résultats :

```
awscurl "https://your-neptune-endpoint:port/ml/modeltransform?maxItems=3" \
```

```
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -s "https://your-neptune-endpoint:port/ml/modeltransform" | python -m json.tool
```

Ou encore :

```
curl -s "https://your-neptune-endpoint:port/ml/modeltransform?maxItems=3" | python -
m json.tool
```

Paramètres pour répertorier les tâches **`modeltransform`**

- **`maxItems`** : (facultatif) nombre maximal d'éléments à renvoyer.

  Type : entier. Par défaut : `10`. Valeur maximale autorisée : `1024`.
- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM qui permet à Neptune d'accéder SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres de votre cluster de bases de données, sinon une erreur se produira.

Gestion des points de terminaison d'inférence à l'aide de la commande **`endpoints`**

Vous utilisez la commande Neptune ML `endpoints` pour créer un point de terminaison d'inférence, vérifier son statut, le supprimer ou répertorier les points de terminaison d'inférence existants.

Création d'un point de terminaison d'inférence à l'aide de la commande Neptune ML **`endpoints`**

Une commande Neptune ML `endpoints` permettant de créer un point de terminaison d'inférence à partir d'un modèle créé par une tâche d'entraînement ressemble à ceci :

AWS CLI

```
aws neptunedata create-ml-endpoint \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique ID for the new endpoint)" \
  --ml-model-training-job-id "(the model-training job-id of a completed job)"
```

Pour plus d'informations, consultez create-ml-endpointle manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.create_ml_endpoint(
    id='(a unique ID for the new endpoint)',
    mlModelTrainingJobId='(the model-training job-id of a completed job)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/endpoints \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/endpoints \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Une commande Neptune ML `endpoints` permettant de mettre à jour un point de terminaison d'inférence existant à partir d'un modèle créé par une tâche d'entraînement ressemble à ceci :

AWS CLI

```
aws neptunedata create-ml-endpoint \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique ID for the new endpoint)" \
  --update \
  --ml-model-training-job-id "(the model-training job-id of a completed job)"
```

Pour plus d'informations, consultez create-ml-endpointle manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.create_ml_endpoint(
    id='(a unique ID for the new endpoint)',
    update=True,
    mlModelTrainingJobId='(the model-training job-id of a completed job)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/endpoints \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "update" : "true",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/endpoints \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "update" : "true",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Une commande Neptune ML `endpoints` permettant de créer un point de terminaison d'inférence à partir d'un modèle créé par une tâche de transformation de modèle ressemble à ceci :

AWS CLI

```
aws neptunedata create-ml-endpoint \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique ID for the new endpoint)" \
  --ml-model-transform-job-id "(the model-transform job-id of a completed job)"
```

Pour plus d'informations, consultez create-ml-endpointle manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.create_ml_endpoint(
    id='(a unique ID for the new endpoint)',
    mlModelTransformJobId='(the model-transform job-id of a completed job)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/endpoints \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTransformJobId": "(the model-transform job-id of a completed job)"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/endpoints \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTransformJobId": "(the model-transform job-id of a completed job)"
      }'
```

Une commande Neptune ML `endpoints` permettant de mettre à jour un point de terminaison d'inférence existant à partir d'un modèle créé par une tâche de transformation de modèle ressemble à ceci :

AWS CLI

```
aws neptunedata create-ml-endpoint \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(a unique ID for the new endpoint)" \
  --update \
  --ml-model-transform-job-id "(the model-transform job-id of a completed job)"
```

Pour plus d'informations, consultez create-ml-endpointle manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.create_ml_endpoint(
    id='(a unique ID for the new endpoint)',
    update=True,
    mlModelTransformJobId='(the model-transform job-id of a completed job)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/endpoints \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "update" : "true",
        "mlModelTransformJobId": "(the model-transform job-id of a completed job)"
      }'
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl \
  -X POST https://your-neptune-endpoint:port/ml/endpoints \
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "update" : "true",
        "mlModelTransformJobId": "(the model-transform job-id of a completed job)"
      }'
```

Paramètres de création d'un point de terminaison d'inférence **`endpoints`**

- **`id`** : (facultatif) identifiant unique du nouveau point de terminaison d'inférence.

  Type : string. Valeur par défaut : nom horodaté généré automatiquement.
- **`mlModelTrainingJobId`** – ID de tâche de la tâche d'entraînement de modèle terminée qui a créé le modèle vers lequel le point de terminaison d'inférence pointera.

  Type : string.

  Remarque : Vous devez fournir `mlModelTrainingJobId` ou `mlModelTransformJobId`.
- **`mlModelTransformJobId`** – ID de tâche de la tâche de transformation de modèle terminée.

  Type : string.

  Remarque : Vous devez fournir `mlModelTrainingJobId` ou `mlModelTransformJobId`.
- **`update`** : (facultatif) s'il est présent, ce paramètre indique qu'il s'agit d'une demande de mise à jour.

  Type : booléen. Par défaut : `false`

  Remarque : Vous devez fournir `mlModelTrainingJobId` ou `mlModelTransformJobId`.
- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM fournissant à Neptune l'accès SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres du cluster de bases de données, sans quoi une erreur sera générée.

- **`modelName`** : (facultatif) type de modèle pour l'entraînement. Par défaut, le modèle ML est automatiquement basé sur le `modelType` utilisé dans le cadre du traitement de données, mais vous pouvez spécifier ici un autre type de modèle.

  Type : string. Valeur par défaut : `rgcn` pour les graphes hétérogènes et `kge` pour les graphes de connaissances. Valeurs valides : pour les graphes hétérogènes : `rgcn`. Pour les graphes de connaissances : `kge`, `transe`, `distmult` ou `rotate`.
- **`instanceType`** : (facultatif) type d'instance ML utilisé pour la maintenance en ligne.

  Type : string. Par défaut : `ml.m5.xlarge`.

  Remarque : Le choix de l'instance ML pour un point de terminaison d'inférence dépend du type de tâche, de la taille de graphe et de votre budget. Consultez Sélection d'une instance pour un point de terminaison d'inférence.
- **`instanceCount`** : (facultatif) nombre minimal d'instances Amazon EC2 à déployer sur un point de terminaison à des fins de prédiction.

  Type : entier. Par défaut : 1.
- **`volumeEncryptionKMSKey`**— (Facultatif) La clé AWS Key Management Service (AWS KMS) utilisée par SageMaker AI pour chiffrer les données sur le volume de stockage attaché aux instances de calcul ML qui exécutent les points de terminaison.

  Type : string. Valeur par défaut : aucune.

Obtention du statut d'un point de terminaison d'inférence à l'aide de la commande Neptune ML **`endpoints`**

Voici un exemple de commande Neptune ML `endpoints` pour obtenir le statut d'un point de terminaison d'instance :

AWS CLI

```
aws neptunedata get-ml-endpoint \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the inference endpoint ID)"
```

Pour plus d'informations, consultez get-ml-endpointle manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.get_ml_endpoint(
    id='(the inference endpoint ID)'
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/endpoints/(the inference endpoint ID)
 \
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -s \
  "https://your-neptune-endpoint:port/ml/endpoints/(the inference endpoint ID)" \
  | python -m json.tool
```

Paramètres d'obtention du statut d'un point de terminaison de l'instance **`endpoints`**

- **`id`** : (obligatoire) identifiant unique du point de terminaison d'inférence.

Type : string.

- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM fournissant à Neptune l'accès SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres du cluster de bases de données, sans quoi une erreur sera générée.

Suppression d'un point de terminaison d'instance à l'aide de la commande Neptune ML **`endpoints`**

Voici un exemple de commande Neptune ML `endpoints` permettant de supprimer un point de terminaison d'instance :

AWS CLI

```
aws neptunedata delete-ml-endpoint \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the inference endpoint ID)"
```

Pour également nettoyer les artefacts associés :

```
aws neptunedata delete-ml-endpoint \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the inference endpoint ID)" \
  --clean
```

Pour plus d'informations, consultez delete-ml-endpointle manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)
```

```
response = client.delete_ml_endpoint(
    id='(the inference endpoint ID)',
    clean=True
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/endpoints/(the inference endpoint ID)
 \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

Pour également nettoyer les artefacts associés :

```
awscurl "https://your-neptune-endpoint:port/ml/endpoints/(the inference endpoint
 ID)?clean=true" \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/endpoints/(the inference endpoint
 ID)"
```

Ou encore :

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/endpoints/(the inference endpoint
 ID)?clean=true"
```

Paramètres de suppression d'un point de terminaison d'inférence **`endpoints`**

- **`id`** : (obligatoire) identifiant unique du point de terminaison d'inférence.

  Type : string.
- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM fournissant à Neptune l'accès SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres du cluster de bases de données, sans quoi une erreur sera générée.
- **`clean`** : (facultatif) indique que tous les artefacts liés à ce point de terminaison doivent également être supprimés.

  Type : booléen. Par défaut : `FALSE`.

Répertorier les points de terminaison d'inférence à l'aide de la commande Neptune ML **`endpoints`**

Une commande Neptune ML `endpoints` pour répertorier les points de terminaison d'inférence ressemble à ceci :

AWS CLI

```
aws neptunedata list-ml-endpoints \
  --endpoint-url https://your-neptune-endpoint:port
```

Pour limiter le nombre de résultats :

```
aws neptunedata list-ml-endpoints \
  --endpoint-url https://your-neptune-endpoint:port \
  --max-items 3
```

Pour plus d'informations, consultez list-ml-endpointsle manuel de référence des AWS CLI commandes.

SDK

```
import boto3
from botocore.config import Config
```

```
client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-neptune-endpoint:port',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.list_ml_endpoints(
    maxItems=3
)

print(response)
```

awscurl

```
awscurl https://your-neptune-endpoint:port/ml/endpoints \
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

Pour limiter le nombre de résultats :

```
awscurl "https://your-neptune-endpoint:port/ml/endpoints?maxItems=3" \
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -s "https://your-neptune-endpoint:port/ml/endpoints" \
  | python -m json.tool
```

Ou encore :

```
curl -s "https://your-neptune-endpoint:port/ml/endpoints?maxItems=3" \
```

```
  | python -m json.tool
```

Paramètres pour répertorier les points de terminaison d'inférence **`dataprocessing`**

- **`maxItems`** : (facultatif) nombre maximal d'éléments à renvoyer.

  Type : entier. Par défaut : `10`. Valeur maximale autorisée : `1024`.
- **`neptuneIamRoleArn`**— (Facultatif) L'ARN d'un rôle IAM fournissant à Neptune l'accès SageMaker aux ressources AI et Amazon S3.

  Type : string. Remarque : Il doit être répertorié dans le groupe de paramètres du cluster de bases de données, sans quoi une erreur sera générée.

Erreurs et exceptions de l'API de gestion Neptune ML

Toutes les exceptions de l'API de gestion Neptune ML renvoient un code HTTP 400. Après avoir reçu l'une de ces exceptions, la commande qui a généré l'exception ne doit pas être réessayée.

- **`MissingParameterException`** – Message d'erreur :

  ```
  Required credentials are missing. Please add IAM role to the cluster or pass as a parameter to this request.
  ```

- **`InvalidParameterException`** – Messages d'erreur :
  - `Invalid ML instance type.`
  - `Invalid ID provided. ID can be 1-48 alphanumeric characters.`
  - `Invalid ID provided. Must contain only letters, digits, or hyphens.`
  - `Invalid ID provided. Please check whether a resource with the given ID exists.`
  - `Another resource with same ID already exists. Please use a new ID.`
  - `Failed to stop the job because it has already completed or failed.`
- **`BadRequestException`** – Messages d'erreur :
  - `Invalid S3 URL or incorrect S3 permissions. Please check your S3 configuration.`
  - `Provided ModelTraining job has not completed.`
  - `Provided SageMaker AI Training job has not completed.`
  - `Provided MLDataProcessing job is not completed.`
  - `Provided MLModelTraining job doesn't exist.`
  - `Provided ModelTransformJob doesn't exist.`
  - `Unable to find SageMaker AI resource. Please check your input.`

Limites de Neptune ML

- Les types d'inférence actuellement pris en charge sont la classification de nœud, la régression de nœud, la classification d'arête, la régression d'arête et la prédiction de lien (voir Fonctionnalités de Neptune ML).
- La taille de graphe maximale que Neptune ML peut prendre en charge dépend de la quantité de mémoire et de stockage requise lors de la préparation des données, de l'entraînement de modèle et de l'inférence.
  - La taille maximale de la mémoire d'une instance de traitement de données SageMaker AI est de 768 Go. Par conséquent, la phase de traitement de données échoue si elle nécessite plus de 768 Go de mémoire.
  - La taille maximale de la mémoire d'une instance de formation SageMaker AI est de 732 Go. Par conséquent, la phase d'entraînement échoue si elle nécessite plus de 732 Go de mémoire.
- La taille maximale d'une charge utile d'inférence pour un point de terminaison SageMaker AI est de 6 MiB. Par conséquent, l'inférence inductive échoue si la charge utile du sous-graphe dépasse cette taille.
- Neptune ML n'est actuellement disponible que dans les régions où Neptune et les autres services dont il dépend (tels qu'Amazon API AWS Lambda Gateway et Amazon SageMaker AI) sont tous pris en charge.

  Il existe des différences dans les régions Chine (Pékin) et Chine (Ningxia) liées à l'utilisation par défaut de l'authentification IAM, comme expliqué ici, entre autres différences.
- Les points de terminaison d'inférence de prédiction de lien lancés par Neptune ML ne peuvent actuellement prédire que les liens possibles avec les nœuds présents dans le graphe au cours de l'entraînement.

  Prenons l'exemple d'un graphe avec les sommets `User` et `Movie`, et les arêtes `Rated`. En utilisant un modèle de recommandation de prédiction de lien Neptune ML correspondant, vous pouvez ajouter un nouvel utilisateur au graphe et demander au modèle de prédire des films pour lui, mais le modèle peut recommander uniquement les films qui étaient présents lors de l'entraînement du modèle. Bien que l'intégration du nœud `User` soit calculée en temps réel à l'aide de son sous-graphe local et du modèle GNN, et qu'elle puisse donc changer au fil du temps, au fur et à mesure que les utilisateurs évaluent les films, elle est comparée aux intégrations de film précalculées statiques pour la recommandation finale.
- Les modèles KGE pris en charge par Neptune ML fonctionnent uniquement pour les tâches de prédiction de lien, et les représentations sont spécifiques aux sommets et aux types d'arête

présents dans le graphe pendant l'entraînement. Cela signifie que tous les sommets et types d'arête auxquels il est fait référence dans une requête d'inférence doivent avoir été présents dans le graphe pendant l'entraînement. Il est impossible d'effectuer des prédictions pour de nouveaux types d'arête ou de nouveaux sommets sans réentraîner le modèle.

SageMaker Limitations des ressources de l'IA

En fonction de vos activités et de l'utilisation des ressources au fil du temps, vous pouvez rencontrer des messages d'erreur indiquant que vous avez dépassé votre quota (ResourceLimitExceeded). Et pour augmenter vos ressources d' SageMaker IA, suivez les étapes de la procédure Demander une augmentation du quota de service pour les SageMaker ressources de cette page pour demander une augmentation de quota au AWS Support.

SageMaker Les noms des ressources AI correspondent aux étapes de Neptune ML comme suit :

- L' SageMaker IA `ProcessingJob` est utilisée par Neptune pour le traitement des données, la formation des modèles et les tâches de transformation de modèles.
- L' SageMaker IA `HyperParameterTuningJob` est utilisée par Neptune Model Training Jobs.
- L' SageMaker IA `TrainingJob` est utilisée par Neptune Model Training Jobs.
- L' SageMaker IA `Endpoint` est utilisée par les points de terminaison d'inférence de Neptune.

Surveillance des ressources Amazon Neptune

Amazon Neptune prend en charge différentes méthodes pour surveiller les performances et l'utilisation des bases de données :

- Statut de l'instance : vérifiez l'état du moteur de base de données orientée graphe d'un cluster Neptune, déterminez la version du moteur qui est installée et obtenez d'autres informations sur l'instance à l'aide de l'API de statut d'instance.
- API de résumé de graphe : l'API de résumé de graphe vous permet de vous faire rapidement une idée de la taille et du contenu des données de votre graphe.

> **Note**
>
> Comme l'API de résumé de graphe repose sur les statistiques DFE, elle n'est disponible que lorsque les statistiques sont activées, ce qui n'est pas le cas sur les types d'instances T3 et T4g.

- Amazon CloudWatch — Neptune envoie automatiquement des métriques aux alarmes CloudWatch et les prend également en charge CloudWatch . Pour de plus amples informations, veuillez consulter the section called “En utilisant CloudWatch”.
- Fichiers journaux d'audit : affichez, téléchargez ou consultez les fichiers journaux de base de données à l'aide de la console Neptune. Pour de plus amples informations, veuillez consulter the section called “Journaux d'audit avec Neptune”.
- Publication de CloudWatch journaux sur Amazon Logs : vous pouvez configurer un cluster de base de données Neptune pour publier les données du journal d'audit dans un groupe de journaux dans Amazon CloudWatch Logs. Avec CloudWatch Logs, vous pouvez effectuer une analyse en temps réel des données du journal, l'utiliser CloudWatch pour créer des alarmes et afficher les métriques, et utiliser CloudWatch les journaux pour stocker vos enregistrements de journal dans un stockage hautement durable. Consultez Logs Neptune CloudWatch .
- AWS CloudTrail— Neptune prend en charge la journalisation des API en utilisant. CloudTrail Pour de plus amples informations, veuillez consulter the section called “Journalisation des appels d'API Neptune avec AWS CloudTrail”.
- Événements : consultez les événements Neptune dans la console ou dans l'API AWS CLI, et abonnez-vous aux notifications d'événements via Amazon SNS. Pour de plus amples informations, veuillez consulter the section called “Événements”.

- Balisage – Utilisez des balises pour ajouter des métadonnées à vos ressources Neptune et suivre l'utilisation en fonction des balises. Pour de plus amples informations, veuillez consulter the section called "Marquage des ressources Neptune".

Rubriques

- Vérification du statut d'une instance Neptune
- Surveillance de Neptune à l'aide d'Amazon CloudWatch
- Utilisation des journaux d'audit avec les cluster Amazon Neptune
- Publication de Neptune Logs sur Amazon Logs CloudWatch
- Activation d'Amazon CloudWatch Logs pour un bloc-notes Neptune
- Utilisation de la journalisation des requêtes lentes Amazon Neptune
- Journalisation des appels d'API Amazon Neptune avec AWS CloudTrail
- Événements Amazon Neptune
- Marquage des ressources Amazon Neptune

Vérification du statut d'une instance Neptune

Amazon Neptune offre un mécanisme permettant de vérifier le statut de la base de données orientée graphe sur l'hôte. Il s'agit également d'une bonne solution pour confirmer que vous êtes en mesure de vous connecter à une instance.

Pour vérifier l'état d'une instance et obtenir l'état du cluster de bases de données à l'aide de `curl` :

```
curl -G https://your-neptune-endpoint:port/status
```

À partir de la version 1.2.1.0.R6 du moteur, vous pouvez utiliser la commande CLI suivante à la place :

```
aws neptunedata get-engine-status
```

Si l'instance est saine, la commande `status` renvoie un objet JSON avec les champs suivants :

- **`status`** : défini sur `"healthy"` si l'instance ne rencontre aucun problème.

Si l'instance est en cours de récupération suite à un incident ou parce qu'elle a été redémarrée et que des transactions actives sont en cours d'exécution depuis le dernier arrêt de serveur, `status` est défini sur `"recovery"`.

- **`startTime`** : défini sur l'heure UTC (heure universelle coordonnée) à laquelle le processus serveur actuel a démarré.
- **`dbEngineVersion`** : défini sur la version de moteur Neptune exécutée sur votre cluster de bases de données.

  Si cette version de moteur a été corrigée manuellement depuis sa publication, le numéro de version est préfixé par `"Patch-"`.
- **`role`** : défini sur `"reader"` si l'instance est un réplica en lecture ou sur `"writer"` s'il s'agit de l'instance principale.
- **`dfeQueryEngine`** : défini sur `"enabled"` si le moteur DFE est complètement activé, ou sur `viaQueryHint` (valeur par défaut) si le moteur DFE n'est utilisé qu'avec les requêtes dont l'indicateur de requête `useDFE` est défini sur `true` (`viaQueryHint` est la valeur par défaut).
- **`gremlin`** : contient des informations sur le langage de requête Gremlin disponible sur votre cluster. Plus précisément, il contient un `version` champ qui indique la TinkerPop version actuelle utilisée par le moteur.
- **`sparql`** : contient des informations sur le langage de requête SPARQL disponible sur votre cluster. Plus précisément, il contient un champ `version` qui spécifie la version actuelle de SPARQL utilisée par le moteur.
- **`opencypher`** : contient des informations sur le langage de requête openCypher disponible sur votre cluster. Plus précisément, il contient un champ `version` qui spécifie la version actuelle d'openCypher utilisée par le moteur.
- **`labMode`** : contient les paramètres Mode Lab utilisés par le moteur. Il s'agit d'une liste sélective de paramètres du mode laboratoire et non de l'ensemble complet. Consultez le groupe de paramètres du cluster pour connaître l'ensemble complet utilisé.
- **`rollingBackTrxCount`** : si des transactions sont annulées, ce champ est défini sur le nombre de transactions de ce type. S'il n'y en a pas, ce champ ne s'affiche pas.
- **`rollingBackTrxEarliestStartTime`** : heure de début de la transaction la plus ancienne annulée. Si aucune transaction n'est annulée, ce champ ne s'affiche pas.
- **`features`** : contient des informations de statut sur les fonctionnalités activées sur votre cluster de bases de données.

- **`lookupCache`** : statut actuel de Cache de recherche. Ce champ n'apparaît que sur les types d'instances `R5d`, car ce sont les seules instances où un cache de recherche peut exister. Ce champ est un objet JSON sous la forme :

  ```
  "lookupCache": {
    "status": "current lookup cache status"
  }
  ```

  Dans une instance `R5d` :

  - Si le cache de recherche est activé, le statut indique `"Available"`.
  - Si le cache de recherche a été désactivé, le statut indique `"Disabled"`.
  - Si la limite de disque a été atteinte sur l'instance, le statut indique `"Read Only Mode - Storage Limit Reached"`.

- **`ResultCache`** : statut actuel de Mise en cache des résultats de requête. Ce champ est un objet JSON sous la forme :

  ```
  "ResultCache": {
    "status": "current results cache status"
  }
  ```

  - Si le cache des résultats a été activé, le statut indique `"Available"`.
  - Si le cache est désactivé, le statut indique `"Disabled"`.

- **`IAMAuthentication`**— Spécifie si l'authentification Gestion des identités et des accès AWS (IAM) a été activée ou non sur votre cluster de base de données :
  - Si l'authentification IAM a été activée, le statut indique `"enabled"`.
  - Si l'authentification IAM est désactivée, le statut indique `"disabled"`.
- **`Streams`** : spécifie si les flux Neptune ont été activés ou non sur le cluster de bases de données :
  - Si les flux sont activés, le statut indique `"enabled"`.
  - Si les flux sont désactivés, le statut indique `"disabled"`.
- **`AuditLog`** : égal à `enabled` si les journaux d'audit sont activés. `disabled`, dans le cas contraire.
- **`SlowQueryLogs`** : égal à `info` ou `debug` si la journalisation des requêtes lentes est activée. `disabled`, dans le cas contraire.

  - **`QueryTimeout`** : valeur, en millisecondes, du délai d'expiration des requêtes.
- **`settings`** : paramètres appliqués à l'instance :
  - **`clusterQueryTimeoutInMs`** : valeur, en millisecondes, du délai d'expiration de la requête, défini pour l'ensemble du cluster.
  - **`SlowQueryLogsThreshold`** : valeur, en millisecondes, du délai d'expiration de la requête, défini pour l'ensemble du cluster.
- **`serverlessConfiguration`** : paramètres sans serveur pour un cluster s'il fonctionne en mode sans serveur :
  - **`minCapacity`** : plus petite taille possible pour une instance sans serveur de votre cluster de bases de données, en unités de capacité Neptune (NCU).
  - **`maxCapacity`** : plus grande taille possible pour une instance sans serveur de votre cluster de bases de données, en unités de capacité Neptune (NCU).

Exemple de sortie de la commande instance status

Voici un exemple du résultat de la commande de statut d'instance (dans ce cas, exécutée sur une instance R5d) :

```
{
  'status': 'healthy',
  'startTime': 'Thu Aug 24 21:47:12 UTC 2023',
  'dbEngineVersion': '1.2.1.0.R4',
  'role': 'writer',
  'dfeQueryEngine': 'viaQueryHint',
  'gremlin': {'version': 'tinkerpop-3.6.2'},
  'sparql': {'version': 'sparql-1.1'},
  'opencypher': {'version': 'Neptune-9.0.20190305-1.0'},
  'labMode': {
    'ObjectIndex': 'disabled',
    'ReadWriteConflictDetection': 'enabled'
  },
  'features': {
    'SlowQueryLogs': 'disabled',
    'ResultCache': {'status': 'disabled'},
    'IAMAuthentication': 'disabled',
    'Streams': 'disabled',
    'AuditLog': 'disabled'
  },
```

```
  'settings': {
    'clusterQueryTimeoutInMs': '120000',
    'SlowQueryLogsThreshold': '5000'
  },
  'serverlessConfiguration': {
    'minCapacity': '1.0',
    'maxCapacity': '128.0'
  }
}
```

S'il y a un problème avec l'instance, la commande de demande de statut renvoie le code d'erreur `HTTP 500`. Si l'hôte est inaccessible, la demande expire. Assurez-vous que vous accédez à l'instance depuis le Virtual Private Cloud (VPC) et que vos groupes de sécurité vous autorisent l'accès à celui-ci.

Surveillance de Neptune à l'aide d'Amazon CloudWatch

Amazon Neptune et Amazon CloudWatch sont intégrés afin que vous puissiez recueillir et analyser les indicateurs de performance. Vous pouvez surveiller ces métriques à l'aide de la CloudWatch console, du AWS Command Line Interface (AWS CLI) ou de l' CloudWatch API.

CloudWatch vous permet également de définir des alarmes afin d'être averti si une valeur métrique dépasse un seuil que vous spécifiez. Vous pouvez même configurer CloudWatch des événements pour prendre des mesures correctives en cas de violation. Pour plus d'informations sur l'utilisation CloudWatch et les alarmes, consultez la CloudWatch documentation.

Rubriques

- Affichage CloudWatch des données (console)
- Affichage CloudWatch des données (AWS CLI)
- Affichage CloudWatch des données (API)
- Utilisation CloudWatch pour surveiller les performances des instances de base de données dans Neptune
- Métriques Neptune CloudWatch
- Dimensions de Neptune CloudWatch

Affichage CloudWatch des données (console)

Pour afficher CloudWatch les données d'un cluster Neptune (console)

1. Connectez-vous à la CloudWatch console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans le panneau de navigation, sélectionnez Métriques.
3. Dans le volet Toutes les mesures, choisissez Neptune, puis choisissez. DBClusterIdentifier
4. Dans le volet supérieur, faites défiler l'écran pour afficher la liste complète des métriques de votre cluster. Les options de métrique Neptune disponibles apparaissent dans la liste Affichage.

Pour sélectionner ou désélectionner une métrique individuelle, sélectionnez la case à côté du métrique et du nom de ressource dans le volet de résultats. Les graphes illustrant les métriques des éléments sélectionnés s'affichent en bas de la console. Pour en savoir plus sur les CloudWatch graphiques, consultez Graph Metrics dans le guide de CloudWatch l'utilisateur Amazon.

Affichage CloudWatch des données (AWS CLI)

Pour afficher CloudWatch les données d'un cluster Neptune (AWS CLI)

1. Installez le AWS CLI. Pour obtenir des instructions, consultez le Guide de l'utilisateur AWS Command Line Interface.
2. Utilisez le AWS CLI pour récupérer des informations. Les CloudWatch paramètres pertinents pour Neptune sont répertoriés dans. Métriques Neptune CloudWatch

   L'exemple suivant récupère CloudWatch les métriques relatives au nombre de requêtes G705 par seconde pour le `gremlin-cluster` cluster.

```
<![CDATA[
aws cloudwatch get-metric-statistics \
    --namespace AWS/Neptune  --metric-name GremlinRequestsPerSec \
    --dimensions Name=DBClusterIdentifier,Value=gremlin-cluster \
    --start-time 2018-03-03T00:00:00Z --end-time 2018-03-04T00:00:00Z \
    --period 60 --statistics=Average
]]>
```

Affichage CloudWatch des données (API)

CloudWatch prend également en charge une `Query` action qui vous permet de demander des informations par programmation. Pour plus d'informations, consultez la documentation de l'API CloudWatch Query et le manuel Amazon CloudWatch API Reference.

Lorsqu'une CloudWatch action nécessite un paramètre spécifique à la surveillance de Neptune, par exemple`MetricName`, utilisez les valeurs répertoriées dans. Métriques Neptune CloudWatch

L'exemple suivant montre une CloudWatch demande de bas niveau utilisant les paramètres suivants :

- `Statistics.member.1 = Average`
- `Dimensions.member.1 = DBClusterIdentifier=gremlin-cluster`
- `Namespace = AWS/Neptune`
- `StartTime = 2013-11-14T00:00:00Z`
- `EndTime = 2013-11-16T00:00:00Z`
- `Period = 60`
- `MetricName = GremlinRequestsPerSec`

Voici à quoi ressemble la CloudWatch demande. Cependant, ceci est juste pour montrer la forme de la requête. Vous devez construire votre propre requête en fonction de vos métriques et période.

```
https://monitoring.amazonaws.com/
      ?SignatureVersion=2
      &Action=GremlinRequestsPerSec
      &Version=2010-08-01
      &StartTime=2018-03-03T00:00:00
      &EndTime=2018-03-04T00:00:00
      &Period=60
      &Statistics.member.1=Average
      &Dimensions.member.1=DBClusterIdentifier=gremlin-cluster
      &Namespace=AWS/Neptune
      &MetricName=GremlinRequests
      &Timestamp=2018-03-04T17%3A48%3A21.746Z
      &AWSAccessKeyId=AWS Access Key ID;
      &Signature=signature
```

Utilisation CloudWatch pour surveiller les performances des instances de base de données dans Neptune

Vous pouvez utiliser CloudWatch les métriques de Neptune pour surveiller ce qui se passe sur vos instances de base de données et suivre la longueur de la file d'attente des requêtes telle qu'observée par la base de données. Les métriques suivantes sont particulièrement utiles :

- **`CPUUtilization`** : pourcentage d'utilisation de CPU.
- **`VolumeWriteIOPs`**— Indique le nombre moyen d' I/O écritures sur disque sur le volume du cluster, indiqué à intervalles de 5 minutes.
- **`MainRequestQueuePendingRequests`** : affiche le nombre de requêtes en attente dans l'exécution de la file d'attente en entrée.

Vous pouvez également déterminer combien de demandes sont en attente sur le serveur en utilisant le point de terminaison de statut des requêtes Gremlin avec le paramètre `includeWaiting`. Indique le statut de toutes les requêtes en attente.

Les indicateurs suivants peuvent vous aider à ajuster vos stratégies de provisionnement et de requêtes Neptune afin d'améliorer l'efficacité et les performances :

- La latence constante, élevée`CPUUtilization`, élevée `VolumeWriteIOPs` et faible `MainRequestQueuePendingRequests` réunies indiquent que le serveur est activement engagé dans le traitement des demandes d'écriture simultanées à un rythme soutenu, avec peu d' I/O attente.
- En revanche, une latence constante, des valeurs `CPUUtilization` et `VolumeWriteIOPs` faibles ainsi qu'une valeur `MainRequestQueuePendingRequests` nulle indiquent que vous avez plus de capacité que nécessaire sur l'instance de base de données principale pour le traitement des demandes d'écriture.
- Des valeurs`CPUUtilization` et `VolumeWriteIOPs` élevées, mais une latence et une valeur `MainRequestQueuePendingRequests` variables indiquent que vous envoyez plus de tâches que ce que le serveur est en mesure de traiter dans un intervalle donné. Envisagez de créer ou de redimensionner des demandes par lots afin d'effectuer la même quantité de travail avec moins de frais transactionnels, en and/or redimensionnant l'instance principale pour augmenter le nombre de fils de requêtes capables de traiter les demandes d'écriture simultanément.
- Une valeur faible `CPUUtilization` associée à une valeur élevée `VolumeWriteIOPs` signifie que les threads de requête attendent la fin I/O des opérations sur la couche de

stockage. Si vous constatez des latences variables et une certaine augmentation de la valeur `MainRequestQueuePendingRequests`, envisagez de redimensionner les demandes par lots ou d'en créer afin d'effectuer le même volume de travail avec moins de charge transactionnelle.

Métriques Neptune CloudWatch

> **Note**
>
> Amazon Neptune envoie des métriques CloudWatch uniquement lorsqu'elles ont une valeur différente de zéro.
> Pour toutes les métriques Neptune, la granularité d'agrégation est de cinq minutes.

Rubriques

- Métriques Neptune CloudWatch
- CloudWatch Métriques désormais obsolètes dans Neptune

Métriques Neptune CloudWatch

Le tableau suivant répertorie les CloudWatch mesures prises en charge par Neptune.

> **Note**
>
> Toutes les métriques cumulées sont réinitialisées chaque fois que le serveur redémarre, que ce soit pour des raisons de maintenance, de redémarrage ou de reprise après un incident.

Métriques Neptune CloudWatch

Métrique	Description	Time Interval	Statistique de l'instance
`BackupRetentionPeriodStorageUsed`	Quantité totale de stockage de sauvegarde, en octets, utilisée pour prendre en charge la		

Métrique	Description	Time Interval	Statistique de l'instance
	fenêtre de rétention de sauvegarde du cluster de bases de données Neptune. Incluse dans le total indiqué par la métrique `TotalBackupStorageBilled` .		
`BufferCacheHitRatio`	Pourcentage de demandes traitées par le cache de tampon. Cette métrique peut être utile pour diagnostiquer la latence des requêtes, car les erreurs de cache induisent une latence importante. Si le taux d'accès au cache est inférieur à 99,9 %, envisagez de mettre à niveau le type d'instance afin de mettre en cache davantage de données en mémoire.	1 minute	average

Métrique	Description	Time Interval	Statistique de l'instance
`ClusterReplicaLag`	Pour un réplica en lecture, durée du retard lors de la réplication des mises à jour à partir de l'instance principale, en millisecondes.	1 minute	average
`ClusterReplicaLagMaximum`	Durée maximale du retard entre l'instance principale et chaque instance de base de données Neptune du cluster de bases de données, en millisecondes.	1 minute	max/min
`ClusterReplicaLagMinimum`	Durée minimale du retard entre l'instance principale et chaque instance de base de données Neptune du cluster de bases de données, en millisecondes.	1 minute	max/min

Métrique	Description	Time Interval	Statistique de l'instance
`CPUCredit Balance`	Nombre de crédits CPU accumulés par une instance, rapporté, rapporté par intervalles de 5 minutes. Cette métrique vous permet de déterminer combien de temps une instance de base de données peut fonctionner en rafale au-delà de son niveau de performance de départ à un débit donné.	5 min	average
`CPUCreditUsage`	Nombre de crédits CPU consommés au cours de la période spécifiée, rapporté par intervalles de 5 minutes. Cette métrique mesure le laps de temps au cours duquel des processeurs physiques ont été utilisées pour traiter les instructions de processeur virtuelles allouées à l'instance de base de données.	5 min	average

Métrique	Description	Time Interval	Statistique de l'instance
`CPUSurplusCreditBalance`	Nombre de crédits excédentaires dépensés par une instance illimitée lorsque la valeur `CPUCreditBalance` est nulle. La valeur de `CPUSurplusCreditBalance` est remboursée progressivement par les crédits UC gagnés. Si le nombre de crédits excédentaires dépasse le nombre maximum de crédits que l'instance peut gagner en 24 heures, les crédits excédentaires dépensés au-dessus du maximum génèrent des frais supplémentaires. Les métriques de crédits CPU sont disponibles toutes les 5 minutes uniquement.	5 min	average

Métrique	Description	Time Interval	Statistique de l'instance
`CPUSurplusCreditsCharged`	Le nombre de crédits excédentaires dépensés qui ne sont pas remboursés par les crédits CPU gagnés et qui entraînent des frais supplémentaires.	5 min	average
`CPUUtilization`	Pourcentage d'utilisation de la CPU.	1 minute	average/P99
`EngineUptime`	Temps d'exécution de l'instance, en secondes.	1 min	average
`FreeableMemory`	Quantité de mémoire vive disponible, en octets.	1 min	average
`GlobalDbDataTransferBytes`	Nombre d'octets de données de journalisation transférés de la base de données principale Région AWS à la base secondaire Région AWS dans une base de données globale Neptune.	1 min	average

Métrique	Description	Time Interval	Statistique de l'instance
`GlobalDbReplicatedWriteIO`	Nombre d' I/O opérations d'écriture répliquées depuis le volume principal Région AWS de la base de données globale vers le volume de cluster d'un volume secondaire Région AWS. Les calculs de facturation pour chaque cluster de bases de données dans une base de données globale Neptune utilisent la métrique `VolumeWriteIOPS` pour prendre en compte les écritures effectuées dans ce cluster. Pour le cluster de bases de données principal, les calculs de facturation utilisent `GlobalDbReplicatedWriteIO` afin de prendre en compte la réplication entre régions vers les	5 min	sum

Métrique	Description	Time Interval	Statistique de l'instance
	clusters de bases de données secondaires.		
`GlobalDbProgressLag`	Retard du cluster secondaire en millisecondes par rapport au cluster principal pour les transactions utilisateur et système.	1 min	average
`GremlinClientErrorsPerSec`	Nombre d'erreurs côté client par seconde dans les traversées Gremlin.	1 min	average
`GremlinServerErrorsPerSec`	Nombre d'erreurs côté serveur par seconde dans les traversées Gremlin.	1 min	average
`GremlinRequestsPerSec`	Nombre de demandes par seconde adressées au moteur Gremlin.	1 min	average
`GremlinWebSocketOpenConnections`	Le nombre de WebSocket connexions ouvertes à Neptune.	1 min	sum
`LoaderClientErrorsPerSec`	Nombre d'erreurs côté client par seconde à partir des demandes du chargeur.	1 min	average

Métrique	Description	Time Interval	Statistique de l'instance
`LoaderRequestsPerSec`	Nombre de demandes du chargeur par seconde.	1 min	average
`LoaderServerErrorsPerSec`	Nombre d'erreurs côté serveur du chargeur par seconde.	1 min	average
`MainRequestQueuePendingRequests`	Nombre de requêtes en attente dans l'exécution de la file d'attente en entrée. Neptune commence à limiter les demandes lorsqu'elles dépassent la capacité maximale de la file d'attente.	1 min	sum

Métrique	Description	Time Interval	Statistique de l'instance
`NCUUtilization`	Applicable uniquement à une instance de base de données ou à un cluster de bases de données Neptune sans serveur. Au niveau de l'instance, indique un pourcentage calculé comme le nombre d'unités de capacité Neptune (NCU) actuellement utilisées par l'instance en question, divisé par le paramètre de capacité NCU maximale du cluster. Une NCU, ou unité de capacité Neptune, se compose de 2 Gio (gibioctet) de mémoire (RAM), en plus de la capacité du processeur virtuel (vCPU) et du réseau associés. Au niveau du cluster, `NCUUtilization` indique le pourcentage de capacité maximale utilisée par		

Métrique	Description	Time Interval	Statistique de l'instance
	le cluster dans son ensemble.		
`NetworkReceiveThroughput`	Le débit réseau entrant reçu des clients par chaque instance du cluster de base de données Neptune, en octets par seconde. Ce débit n'inclut pas le trafic réseau entre les instances du cluster de bases de données et le volume de cluster.	1 min	average
`NetworkThroughput`	Quantité de débit réseau reçue des clients et transmise à ces derniers par chaque instance du cluster de bases de données Neptune, en octets par seconde. Ce débit n'inclut pas le trafic réseau entre les instances du cluster de bases de données et le volume de cluster.	1 min	average

Métrique	Description	Time Interval	Statistique de l'instance
`NetworkTransmitThroughput`	Quantité de débit réseau sortant transmise aux clients par chaque instance du cluster de bases de données Neptune, en octets par seconde. Ce débit n'inclut pas le trafic réseau entre les instances du cluster de bases de données et le volume de cluster.	1 min	average
`NumIndexDeletesPerSec`	Nombre de suppressions dans des index individuels. Les suppressions de chaque index sont comptées individuellement. Cela inclut les suppressions qui peuvent être annulées si une requête rencontre une erreur.	1 min	average

Métrique	Description	Time Interval	Statistique de l'instance
`NumIndexInsertsPerSec`	Nombre d'encarts dans les index individuels. Les insertions dans chaque index sont comptées séparément. Cela inclut les insertions qui peuvent être annulées si une requête rencontre une erreur.	1 min	average
`NumIndexReadsPerSec`	Nombre de relevés scannés à partir de n'importe quel index. Tout modèle d'accès commence par une recherche dans un index et la lecture de toutes les instructions correspondantes. Une augmentation de cette métrique peut entraîner une augmentation de la latence des requêtes ou de l'utilisation du processeur.	1 min	average
`NumQueuedRequestsPerSec`	Le nombre de demandes mises en file d'attente par seconde.		

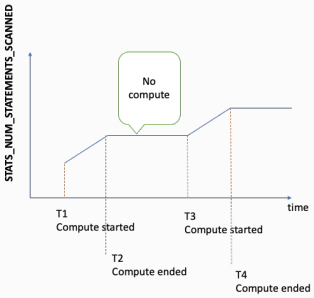
Métrique	Description	Time Interval	Statistique de l'instance
`NumResultCacheHit`	Nombre de visites dans le cache des résultats de Gremlin.	1 min	sum
`NumResultCacheMiss`	Nombre de caches de résultats manquants dans le cache de résultats de Gremlin.	1 min	sum
`NumTxCommitted`	Nombre de transactions validées par seconde avec succès.	1 min	sum
`NumTxOpened`	Nombre de transactions ouvertes sur le serveur par seconde.	1 min	sum
`NumTxRolledBack`	Pour les requêtes d'écriture, nombre de transactions par seconde restaurées sur le serveur en raison d'erreurs. Pour les requêtes en lecture seule, cette métrique est égale au nombre de transactions en lecture seule effectuées par seconde.	1 min	sum

Métrique	Description	Time Interval	Statistique de l'instance
NumUndoPagesPurged	Cette métrique indique le nombre de lots purgés. Cette métrique est un indicateur des progrès réalisés en matière de purge. La valeur concerne 0 les instances de lecteur et la métrique ne s'applique qu'à l'instance d'écriture.	1 min	sum
OpenCypherRequestsPerSec	Nombre de demandes par seconde (HTTPS et Bolt) adressées au moteur openCypher.	1 min	average
OpenCypherBoltOpenConnections	Nombre de connexions WebSocket ouvertes avec Bolt.	1 min	sum
ResultCacheSizeInBytes	Taille totale estimée (en octets) de tous les éléments mis en cache dans le cache de résultats G705.	1 min	sum
ResultCacheItemCount	Nombre d'éléments dans le cache de résultats de Gremlin.	1 min	sum

Métrique	Description	Time Interval	Statistique de l'instance
`ResultCacheOldestItemTimestamp`	Horodatage de l'élément le plus ancien mis en cache dans le cache de résultats de Gremlin.	1 min	sum
`ResultCacheNewestItemTimestamp`	Horodatage du dernier élément mis en cache dans le cache de résultats de Gremlin.		

Métrique	Description	Time Interval	Statistique de l'instance
`ServerlessDatabaseCapacity`	En tant que métrique de niveau instance, `ServerlessDatabaseCapacity` indique la capacité actuelle d'une instance Neptune sans serveur donnée, en NCU. Une NCU, ou unité de capacité Neptune, se compose de 2 Gio (gibioctet) de mémoire (RAM), en plus de la capacité du processeur virtuel (vCPU) et du réseau associés. Au niveau d'un cluster, `ServerlessDatabaseCapacity` représente la moyenne des valeurs `ServerlessDatabaseCapacity` de toutes les instances de base de données du cluster.		

Métrique	Description	Time Interval	Statistique de l'instance
`SnapshotStorageUsed`	Quantité totale de stockage de sauvegarde en octets consommée par tous les instantanés pour un cluster de bases de données Neptune en dehors de sa période de rétention des sauvegardes. Incluse dans le total indiqué par la métrique `TotalBackupStorageBilled` .	1 min	sum
`SparqlClientErrorsPerSec`	Nombre d'erreurs côté client par seconde dans les requêtes SPARQL.	1 min	average
`SparqlRequestsPerSec`	Nombre de demandes envoyées au moteur SPARQL par seconde.	1 min	average
`SparqlServerErrorsPerSec`	Le nombre d'erreurs du serveur SPARQL par seconde.	1 min	average

Métrique	Description	Time Interval	Statistique de l'instance
`StatsNumStatementsScanned`	Nombre total de déclarations analysées pour les statistiques DFE depuis le démarrage du serveur. Chaque fois que le calcul des statistiques est déclenché, ce nombre augmente. Quand aucun calcul n'est effectué, il reste constant. Par conséquent, si vous le représentez sous forme de graphe au fil du temps, vous savez quand le calcul a eu lieu et quand il n'a pas eu lieu :  En examinant la pente du graphe aux périodes où la métrique augmente, vous pouvez	1 min	sum

Métrique	Description	Time Interval	Statistique de l'instance
	également déterminer la rapidité du calcul. Si aucune métrique de ce type n'existe, cela signifie que la fonctionnalité de statistiques est désactivée sur le cluster de bases de données ou que la version du moteur que vous exécutez ne dispose pas de la fonctionnalité correspondante. Si la valeur de la métrique est nulle, cela signifie qu'aucun calcul de statistiques n'a eu lieu.		
`StorageNetworkReceiveThroughput`	Le débit réseau reçu du sous-système de stockage par chaque instance du cluster de base de données Neptune.	1 min	average

Métrique	Description	Time Interval	Statistique de l'instance
`StorageNetworkThroughput`	Quantité de débit réseau reçue et envoyée au sous-système de stockage par chaque instance du cluster de base de données Neptune.	1 min	average
`StorageNetworkTransmitThroughput`	Quantité de débit réseau envoyée au sous-système de stockage par chaque instance du cluster de base de données Neptune.	1 min	average
`SwapUsage`	Quantité d'espace d'échange utilisé.	1 min	sum
`TempStorageIOPS`	Le nombre d'IOPS en lecture et en écriture sur le stockage local attaché à l'instance de base de données Neptune. Cette métrique représente un nombre et est mesurée une fois par seconde.		

Métrique	Description	Time Interval	Statistique de l'instance
`TempStorageThroughput`	La quantité de données transférées vers et depuis le stockage local associé à l'instance de base de données Neptune. Cette métrique représente des octets et est mesurée une fois par seconde.		
`TotalBackupStorageBilled`	Quantité totale de stockage de sauvegarde en octets qui vous est facturée pour un cluster de bases de données Neptune spécifique. Inclut le stockage de sauvegarde mesuré par les métriques `BackupRetentionPeriodStorageUsed` et `SnapshotStorageUsed` .	1 jour	sum
`TotalRequestsPerSec`	Nombre total de requêtes par seconde adressées au serveur à partir de toutes les sources.	1 min	average

Métrique	Description	Time Interval	Statistique de l'instance
`TotalClientErrorsPerSec`	Nombre total de requêtes par seconde ayant commis une erreur en raison de problèmes côté client.	1 min	average
`TotalServerErrorsPerSec`	Nombre total de requêtes par seconde ayant provoqué une erreur sur le serveur en raison de défaillances internes.	1 min	average

Métrique	Description	Time Interval	Statistique de l'instance
`UndoLogListSize`	Nombre de journaux d'annulation figurant dans la liste des journaux d'annulation. Les journaux d'annulation contiennent des enregistrements de transactions validées qui expirent lorsque toutes les transacti ons actives sont plus récentes que la date de validation. Les enregistrements ayant expiré sont régulière ment purgés. La purge des enregistr ements de suppressi on peut prendre plus de temps que celle des enregistrements relatifs aux autres types de transactions. La purge étant effectuée exclusive ment par l'instanc e d'enregistreur du cluster de bases de données, le taux de purge dépend du type d'instance	1 min	sum

Métrique	Description	Time Interval	Statistique de l'instance
	d'enregistreur. Si la valeur `UndoLogListSize` est élevée et augmente dans votre cluster de bases de données, mettez à niveau l'instance d'enregistreur pour augmenter le taux de purge. De même, si vous effectuez une mise à niveau vers une version du moteur `1.2.0.0` ou supérieure à partir d'une version antérieure`1.2.0.0`, assurez-vous d'abord que la `UndoLogListSize` valeur est inférieure à un certain seuil. Dans le cas contraire, le correctif sera annulé et échouera. Les seuils sont basés sur le type d'instance : la limite par défaut est de 40 000 pour les instances 4 x grandes ou plus, et de 10 000 pour les		

Métrique	Description	Time Interval	Statistique de l'instance
	instances inférieures à 4 x large. Si vous tentez de mettre à niveau un cluster dont la `UndoLogListSize` métrique est supérieure à la limite, le processus de correction sera annulé, et un événement expliquant le motif sera visible sur la page des événements du cluster. Ces limites peuvent changer pour des raisons opérationnelles sans avertissement préalable. Étant donné que les versions du moteur `1.2.0.0` et supérieures utilisent un format différent pour les journaux d'annulation, la mise à niveau ne peut commencer qu'une fois que vos journaux d'annulation précédents ont été complètement purgés en dessous du seuil applicable. Pour plus d’informations,		

Métrique	Description	Time Interval	Statistique de l'instance
	consultez Mise à niveau vers la version 1.2.0.0 ou supérieure.		

Métrique	Description	Time Interval	Statistique de l'instance
`VolumeBytesLeftTotal`	Espace disponible restant pour le volume du cluster, en octets. À mesure que le volume du cluster augmente, cette valeur diminue. S’il atteint zéro, le cluster signale une erreur de manque d’espace. Si vous souhaitez détecter si votre cluster de base de données Neptune approche de sa limite de taille, cette valeur est plus simple et plus fiable à surveiller que. `VolumeBytesUsed` `VolumeBytesLeftTotal` prend en compte le stockage utilisé pour le ménage interne et les autres allocations qui n'ont aucune incidence sur votre facturation de stockage. Cette métrique est signalée uniquement par l'instance du	1 min	average

Métrique	Description	Time Interval	Statistique de l'instance
	rédacteur. Pour éviter toute interruption lors des basculements, utilisez la dimension au niveau du cluster (). `DBCluster Identifier`		
`VolumeBytesUsed`	Quantité totale de stockage allouée à votre cluster de bases de données Neptune en octets. Il s'agit de la quantité de stockage pour laquelle vous êtes facturé. Il s'agit de la quantité maximale de stockage allouée à votre cluster DB à tout moment de son existence, et non de la quantité que vous utilisez actuellement (voir Facturation du stockage Neptune).	5 min	sum

Métrique	Description	Time Interval	Statistique de l'instance
`VolumeReadIOPs`	Le nombre total d' I/O opérations de lecture facturées à partir d'un volume de cluster, indiqué à intervalles de 5 minutes. Les opérations de lecture facturées sont calculées au niveau du volume de cluster, regroupées à partir de toutes les instances du cluster de bases de données Neptune, puis rapportées par intervalles de 5 minutes.	5 min	sum
`VolumeWriteIOPs`	Nombre total d' I/O opérations d'écriture sur le disque sur le volume du cluster, indiqué à intervalles de 5 minutes.	5 min	sum

CloudWatch Métriques désormais obsolètes dans Neptune

L'utilisation de ces métriques Neptune est désormais obsolète. Elles sont toujours prises en charge, mais pourraient être éliminées à l'avenir à mesure que des métriques nouvelles et meilleures deviennent disponibles.

Métrique	Description
`GremlinHttp1xx`	Nombre de réponses HTTP 1xx pour le point de terminaison Gremlin par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http1xx`.
`GremlinHttp2xx`	Nombre de réponses HTTP 2xx pour le point de terminaison Gremlin par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http2xx`.
`GremlinHttp4xx`	Nombre d'erreurs HTTP 4xx pour le point de terminaison Gremlin par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http4xx`.
`GremlinHttp5xx`	Nombre d'erreurs HTTP 5xx pour le point de terminaison Gremlin par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http5xx`.
`GremlinErrors`	Nombre d'erreurs dans le parcours Gremlin.
`GremlinRequests`	Nombre de demandes envoyées au moteur Gremlin.
`GremlinWebSocketSuccess`	Nombre de WebSocket connexions réussies au point de terminaison G705 par seconde.
`GremlinWebSocketClientErrors`	Nombre d'erreurs WebSocket client sur le point de terminaison Gremlin par seconde.
`GremlinWebSocketServerErrors`	Nombre d'erreurs de WebSocket serveur sur le point de terminaison Gremlin par seconde.

Métrique	Description
`GremlinWebSocketAvailableConnections`	Nombre de WebSocket connexions potentielles actuellement disponibles.
`Http100`	Nombre de réponses HTTP 100 pour le point de terminaison par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http1xx`.
`Http101`	Nombre de réponses HTTP 101 pour le point de terminaison par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http1xx`.
`Http1xx`	Nombre de réponses HTTP 1xx pour le point de terminaison par seconde.
`Http200`	Nombre de réponses HTTP 200 pour le point de terminaison par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http2xx`.
`Http2xx`	Nombre de réponses HTTP 2xx pour le point de terminaison par seconde.
`Http400`	Nombre d'erreurs HTTP 400 pour le point de terminaison par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http4xx`.
`Http403`	Nombre d'erreurs HTTP 403 pour le point de terminaison par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http4xx`.

Métrique	Description
`Http405`	Nombre d'erreurs HTTP 405 pour le point de terminaison par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http4xx`.
`Http413`	Nombre d'erreurs HTTP 413 pour le point de terminaison par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http4xx`.
`Http429`	Nombre d'erreurs HTTP 429 pour le point de terminaison par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http4xx`.
`Http4xx`	Nombre d'erreurs HTTP 4xx pour le point de terminaison par seconde.
`Http500`	Nombre d'erreurs HTTP 500 pour le point de terminaison par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http5xx`.
`Http501`	Nombre d'erreurs HTTP 501 pour le point de terminaison par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http5xx`.
`Http5xx`	Nombre d'erreurs HTTP 5xx pour le point de terminaison par seconde.
`LoaderErrors`	Nombre d'erreurs depuis des demandes de chargeur.

Métrique	Description
`LoaderRequests`	Nombre de demandes de chargeur.
`SparqlHttp1xx`	Nombre de réponses HTTP 1xx pour le point de terminaison SPARQL par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http1xx`.
`SparqlHttp2xx`	Nombre de réponses HTTP 2xx pour le point de terminaison SPARQL par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http2xx`.
`SparqlHttp4xx`	Nombre d'erreurs HTTP 4xx pour le point de terminaison SPARQL par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http4xx`.
`SparqlHttp5xx`	Nombre d'erreurs HTTP 5xx pour le point de terminaison SPARQL par seconde. Nous vous recommandons d'utiliser plutôt la nouvelle métrique combinée `Http5xx`.
`SparqlErrors`	Nombre d'erreurs dans les requêtes SPARQL.
`SparqlRequests`	Nombre de demandes envoyées au moteur SPARQL.
`StatusErrors`	Nombre d'erreurs depuis le point de terminais on de statut.
`StatusRequests`	Nombre de demandes envoyées au point de terminaison de statut.

Dimensions de Neptune CloudWatch

Les métriques associées à Amazon Neptune sont qualifiées par les valeurs du compte, du nom de graphe ou de l'opération. Vous pouvez utiliser la CloudWatch console Amazon pour récupérer les données Neptune ainsi que toutes les dimensions indiquées dans le tableau suivant.

Dimension	Description
`DBInstanceIdentifier`	Filtre les données que vous demandez pour une instance de base de données spécifique d'un cluster.
`DBClusterIdentifier`	Filtre les données que vous avez demandées pour un cluster de bases de données Neptune spécifique.
`DBClusterIdentifier` , `EngineName`	Filtre les données en fonction du cluster. Le nom du moteur pour toutes les instances Neptune est `neptune`.
`DBClusterIdentifier` , `Role`	Filtre les données que vous demandez pour un cluster de base de données Neptune spécifiqu e, en agrégeant la métrique par rôle d'instanc e (). WRITER/READER Par exemple, vous pouvez regrouper des métriques pour toutes les instances READER qui appartiennent à un cluster.
`DBClusterIdentifier` , `SourceRegion`	Filtre les données en fonction du cluster principal dans la région principale d'une base de données globale.
`DatabaseClass`	Filtre les données que vous demandez pour toutes les instances d'une classe de base de données. Par exemple, vous pouvez regrouper des métriques pour toutes les instances qui appartiennent à la classe de base de données db.r4.large

Dimension	Description
`EngineName`	Le nom du moteur pour toutes les instances Neptune est `neptune`.
`GlobalDbDBClusterIdentifier` , `SecondaryRegion`	Filtre les données en fonction du cluster secondaire d'une base de données globale spécifiée dans une région secondaire.

Utilisation des journaux d'audit avec les cluster Amazon Neptune

Pour effectuer un audit de l'activité de votre cluster de bases de données Amazon Neptune, activez la collection des journaux d'audit en définissant un paramètre de cluster de bases de données. Lorsque les journaux d'audit sont activés, vous pouvez les utiliser pour consigner n'importe quelle combinaison d'événements pris en charge. Vous pouvez consulter ou télécharger les journaux d'audit pour les examiner.

Activation des journaux d'audit Neptune

Utilisez le paramètre `neptune_enable_audit_log` pour activer (`1`) ou désactiver (`0`) les journaux d'audit.

Définissez ce paramètre dans le groupe de paramètres utilisé par votre cluster de bases de données. Vous pouvez utiliser la procédure illustrée ci-dessous Modification d'un groupe de paramètres de cluster de base de données ou d'un groupe de paramètres de base de données pour modifier le paramètre à l'aide de la AWS Management Console commande modify-db-cluster-parameter-group AWS CLI ou de la commande API pour modifier le paramètre par programmation. ModifyDBClusterParameterGroup

Vous devrez redémarrer vos instances de base de données après avoir modifié ce paramètre pour appliquer la modification.

Affichage des journaux d'audit Neptune à l'aide de la console

Vous pouvez consulter et télécharger les journaux d'audit à l'aide de AWS Management Console. Dans la page Instances, choisissez l'instance de base de données pour afficher ses détails, puis faites défiler la page jusqu'à la section Journaux.

Pour télécharger un fichier journal, sélectionnez ce fichier dans la section Journaux, puis choisissez Télécharger.

Détails des journaux d'audit Neptune

Les fichiers journaux sont au UTF-8 format. Les journaux sont écrits dans plusieurs fichiers, dont le nombre varie en fonction de la taille de l'instance. Pour consulter les derniers événements, vous devrez peut-être passer en revue tous les fichiers journaux d'audit.

Les entrées de journal ne sont pas classées par ordre séquentiel. Vous pouvez utiliser la valeur `timestamp` pour les classer.

Les fichiers journaux font l'objet d'une rotation lorsqu'ils atteignent 100 Mo au total. Cette limite n'est pas configurable.

Les fichiers journaux d'audit incluent les informations suivantes séparées par une virgule, en lignes et dans l'ordre suivant :

Champ	Description
Horodatage	L'horodatage Unix pour l'événement consigné avec une précision à la microseconde.
ClientHost	Nom d'hôte ou adresse IP de connexion de l'utilisateur.
ServerHost	Nom d'hôte ou adresse IP de l'instance pour laquelle l'événement est consigné.
ConnectionType	Type de connexion. Il peut s'agit de `Websocket` , `HTTP_POST` , `HTTP_GET` ou `Bolt`.

ARN IAM de l'appelant	ARN de l'utilisateur ou du rôle IAM utilisé pour signer la demande. Vide si l'authentification IAM est désactivée. Son format est :
	`arn:`*`partition`*` :`*`service`*`:`*`region`*`:`*`account`*`:`*`resource`*
	Par exemple :
	`arn:aws:iam::123456789012:user/Anna`
	`arn:aws:sts::123456789012:assumed-role/AWSNeptuneNotebookRole/SageMaker`

Champ	Description
Auth Context	Contient un objet JSON sérialisé qui comporte les informations d'authentification . Le champ `authenticationSucceeded` est `True` si l'utilisateur a été authentifié. Vide si l'authentification IAM est désactivée.
HttpHeader	Informations sur l'en-tête HTTP. Peut contenir une requête. Raccords WebSocket for/Bolt vides.
Charge utile	Requête Gremlin, SPARQL ou openCypher.

Publication de Neptune Logs sur Amazon Logs CloudWatch

Vous pouvez configurer un cluster de base de données Neptune pour publier les données du journal d'audit (and/or requêtes lentes) dans un groupe de journaux dans Amazon Logs. CloudWatch Avec CloudWatch Logs, vous pouvez effectuer une analyse en temps réel des données du journal, puis les utiliser CloudWatch pour créer des alarmes et afficher des métriques. Vous pouvez utiliser CloudWatch les journaux pour stocker vos enregistrements de journal dans un espace de stockage hautement durable.

Pour publier des journaux d'audit dans CloudWatch Logs, les journaux d'audit doivent être explicitement activés (voirActivation des journaux d'audit). De même, pour publier des journaux de requêtes lentes dans CloudWatch Logs, les journaux de requêtes lentes doivent être explicitement activés (voir). Utilisation de la journalisation des requêtes lentes Amazon Neptune

> **Note**
>
> Tenez compte des points suivants :
>
> - Des frais supplémentaires s'appliquent lorsque vous publiez des journaux sur CloudWatch. Consultez la page de CloudWatch tarification pour plus de détails.
> - Vous ne pouvez pas publier de journaux dans CloudWatch Logs pour la région Chine (Pékin) ou Chine (Ningxia).
> - Si l'exportation des données de journaux est désactivée, Neptune ne supprime pas les groupes de journaux ni les flux de journaux existants. Si l'exportation des données de journal est désactivée, les données de journal existantes restent disponibles dans

CloudWatch les journaux, en fonction de la conservation des journaux, et vous devez toujours payer des frais pour les données des journaux d'audit stockées. Vous pouvez supprimer des flux de journaux et des groupes de journaux à l'aide de la console CloudWatch Logs, de l' AWS CLI API Logs ou de l'API CloudWatch Logs.

Utilisation de la console pour publier les journaux Neptune dans des journaux CloudWatch

Pour publier les journaux Neptune dans Logs depuis la console CloudWatch

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le cluster de bases de données Neptune dont vous voulez publier les données de journaux.
4. Pour Actions, choisissez Modifier.
5. Dans la section Exportations de journaux, choisissez les journaux que vous souhaitez commencer à publier dans CloudWatch Logs.
6. Choisissez Continuer, puis Modifier le cluster DB sur la page récapitulative.

Utilisation de la CLI pour publier les journaux d'audit de Neptune dans Logs CloudWatch

Vous pouvez créer un nouveau cluster de base de données qui publie les journaux d'audit dans CloudWatch Logs à l'aide de la AWS CLI `create-db-cluster` commande avec les paramètres suivants :

```
aws neptune create-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["audit"]'
```

Vous pouvez configurer un cluster de base de données existant pour publier des journaux d'audit dans CloudWatch Logs à l'aide de la AWS CLI `modify-db-cluster` commande avec les paramètres suivants :

```
aws neptune modify-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Utilisation de la CLI pour publier les journaux de requêtes lentes de Neptune dans Logs CloudWatch

Vous pouvez également créer un nouveau cluster de base de données qui publie des journaux à requêtes lentes dans Logs à CloudWatch l'aide de la AWS CLI `create-db-cluster` commande avec les paramètres suivants :

```
aws neptune create-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["slowquery"]'
```

De même, vous pouvez configurer un cluster de base de données existant pour publier des journaux à requêtes lentes dans Logs à CloudWatch l'aide de la AWS CLI `modify-db-cluster` commande avec les paramètres suivants :

```
aws neptune modify-db-cluster --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["slowquery"]}'
```

Surveillance des événements du journal Neptune sur Amazon CloudWatch

Après avoir activé les journaux Neptune, vous pouvez surveiller les événements des journaux dans Amazon CloudWatch Logs. Un groupe de journaux est automatiquement créé pour le cluster de bases de données Neptune sous le préfixe suivant, dans lequel *`cluster-name`* représente le nom du cluster de bases de données et *`log_type`* le type de journal :

```
/aws/neptune/cluster-name/log_type
```

Par exemple, si vous configurez la fonction d'exportation de manière à inclure le journal d'audit pour un cluster de bases de données nommé `mydbcluster`, les données du journal sont stockées dans le groupe de journaux `/aws/neptune/mydbcluster/audit`.

Tous les événements de toutes les instances de base de données d'un cluster de bases de données sont transmis en mode push vers un groupe de journaux par l'intermédiaire de différents flux de journaux.

Si un groupe de journaux portant le nom spécifié existe déjà, Neptune utilise ce groupe pour exporter les données de journaux du cluster de bases de données Neptune. Vous pouvez utiliser la configuration automatique, par exemple pour créer des groupes de journaux avec des périodes de conservation des journaux prédéfinies, des filtres métriques et un accès client. AWS CloudFormation Sinon, un nouveau groupe de journaux est automatiquement créé en utilisant la période de conservation des journaux par défaut, Never Expire, dans CloudWatch Logs.

Vous pouvez utiliser la console CloudWatch Logs AWS CLI, ou l'API CloudWatch Logs pour modifier la période de conservation des journaux. Pour plus d'informations sur la modification des périodes de conservation des CloudWatch journaux dans les journaux, voir Conservation des données des journaux des modifications dans CloudWatch les journaux.

Vous pouvez utiliser la console CloudWatch Logs AWS CLI, ou l'API CloudWatch Logs pour rechercher des informations dans les événements du journal d'un cluster de bases de données. Pour plus d'informations sur la recherche et le filtrage des données de journaux, consultez Recherche et filtrage des données de journaux.

Activation d'Amazon CloudWatch Logs pour un bloc-notes Neptune

CloudWatch Les journaux des blocs-notes Neptune sont désactivés par défaut. Procédez comme suit pour les activer, à des fins de débogage ou autre :

Utilisation de AWS Management Console pour activer CloudWatch les journaux pour un bloc-notes Neptune

1. Ouvrez la console Amazon SageMaker AI à l'adresse https://console.aws.amazon.com/sagemaker/.
2. Dans le panneau de navigation de gauche, choisissez Bloc-notes, puis Instances de blocs-notes. Recherchez le nom du bloc-notes Neptune pour lequel vous souhaitez activer les journaux.
3. Accédez à la page de détails en choisissant le nom de l'instance de bloc-notes mentionnée à l'étape ci-dessus.

4. Si l'instance du bloc-notes est en cours d'exécution, sélectionnez le bouton Arrêter en haut à droite de la page de détails du bloc-notes.
5. Sous Autorisations et chiffrement, il existe un champ pour l'ARN du rôle IAM. Sélectionnez le lien dans ce champ pour accéder au rôle IAM de ce bloc-notes.
6. Créez la politique suivante :

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DeleteLogDelivery",
        "logs:Describe*",
        "logs:GetLogDelivery",
        "logs:GetLogEvents",
        "logs:ListLogDeliveries",
        "logs:PutLogEvents",
        "logs:PutResourcePolicy",
        "logs:UpdateLogDelivery"
      ],
      "Resource": "*"
    }
  ]
}
```

7. Enregistrez cette nouvelle politique et attachez-la au rôle IAM à l'étape 4.
8. Sélectionnez Démarrer en haut à droite de la page de détails de l'instance SageMaker AI Notebook.
9. Une fois que les journaux commencent à circuler, vous devriez voir un lien Afficher les journaux sous le champ intitulé Configuration du cycle de vie en bas à gauche de la section Paramètres d'instance du bloc-notes de la page de détails.

Si votre bloc-notes ne démarre pas, un message s'affichera sur la page de détails du bloc-notes sur la console SageMaker AI indiquant que l'instance du bloc-notes a mis plus de 5 minutes à démarrer. Les CloudWatch journaux relatifs à ce problème se trouvent sous le nom :*(your-notebook-name)*`/LifecycleConfigOnStart`.

Consultez Log Amazon SageMaker Events with Amazon CloudWatch pour plus de détails, si nécessaire.

Utilisation de la journalisation des requêtes lentes Amazon Neptune

Il peut être difficile d'identifier, de déboguer et d'optimiser une requête lente. Lorsque la journalisation des requêtes lentes dans Neptune est activée, les attributs de toutes les requêtes de longue durée sont automatiquement journalisés pour faciliter ce processus.

> **Note**
>
> Slow-query la journalisation a été introduite dans la version 1.2.1.0 du moteur Neptune.

Vous activez la journalisation des requêtes lentes à l'aide du paramètre de cluster de bases de données neptune_enable_slow_query_log. Par défaut, ce paramètre est défini sur `disabled`. En le définissant `info` ou en activant la `debug` journalisation des requêtes lentes. Le paramètre `info` journalise quelques attributs utiles de chaque requête lente, tandis que le paramètre `debug` journalise tous les attributs disponibles.

Pour définir le seuil d'une requête considérée comme lente, utilisez le paramètre de cluster de bases de données neptune_slow_query_log_threshold afin de spécifier le nombre de millisecondes au bout desquelles une requête en cours d'exécution sera considérée comme lente et sera enregistrée lorsque la journalisation des requêtes lentes est activée. La valeur par défaut est de 5 000 millisecondes (5 secondes).

Vous pouvez définir ces paramètres de cluster de base de données dans ou à l'aide de la AWS Management Console commande AWS CLI modify-db-cluster-parameter-group ou de la fonction de gestion. ModifyDBClusterParameterGroup

> **Note**
>
> Les paramètres de journalisation des requêtes lentes sont dynamiques, ce qui signifie que la modification de leurs valeurs ne nécessite ni ne provoque le redémarrage de votre cluster de bases de données.

Pour consulter les journaux des requêtes lentes dans le AWS Management Console

Vous pouvez consulter et télécharger les journaux des requêtes lentes dans le AWS Management Console, comme suit :

Sur la page Instances, choisissez une instance de base de données, puis faites défiler la page jusqu'à la section Journaux. Vous pouvez ensuite y sélectionner un fichier journal, puis choisir Télécharger pour le télécharger.

Fichiers générés par la journalisation des requêtes lentes Neptune

Les fichiers journaux générés par la journalisation des requêtes lentes dans Neptune présentent les caractéristiques suivantes :

- Les fichiers sont encodés au format UTF-8.
- Les requêtes et leurs attributs sont journalisés au format JSON.
- Les attributs nuls et vides ne sont pas journalisés, sauf pour les données `queryTime`.
- Les journaux couvrent plusieurs fichiers dont le nombre varie en fonction de la taille de l'instance.
- Les entrées de journal ne sont pas classées par ordre séquentiel. Vous pouvez utiliser leur valeur `timestamp` pour les classer.
- Pour consulter les derniers événements, vous devrez peut-être passer en revue tous les fichiers journaux de requêtes lentes.
- Les fichiers journaux font l'objet d'une rotation lorsqu'ils atteignent 100 Mio au total. Cette limite n'est pas configurable.

Attributs de requête connectés en mode **`info`**

Les attributs suivants sont journalisés pour les requêtes lentes lorsque le paramètre de cluster de bases de données `neptune_enable_slow_query_log` a été défini sur `info` :

Groupe	Attribut	Description
demande ResponseMetadata	`requestId`	ID de demande de la requête.
	`requestType`	Type de demande, tel que HTTP ou WebSocket.
	`responseStatusCode`	Code d'état de la réponse à la requête, comme 200.
	`exceptionClass`	Classe d'exception de l'erreur renvoyée après l'exécution de la requête.
queryStats	`query`	Chaîne de requête.
	`queryFingerprint`	Empreinte digitale de la requête.
	`queryLanguage`	Langage de requête, tel que Gremlin, SPARQL ou openCypher.
memoryStats	`allocatedPermits`	Autorisations allouées à la requête.
	`approximateUsedMemoryBytes`	Mémoire approximative utilisée par la requête pendant l'exécution.
queryTimeStats	`startTime`	Heure de début de la requête (UTC).

Groupe	Attribut	Description
	`overallRunTimeMs`	Durée totale d'exécution de la requête, en millisecondes.
	`parsingTimeMs`	Durée d'analyse de la requête, en millisecondes.
	`waitingTimeMs`	Temps d'attente de Gremlin/SPARQL/openCypher la file d'attente des requêtes, en millisecondes
	`executionTimeMs`	Durée d'exécution de la requête, en millisecondes.
	`serializationTimeMs`	Durée de sérialisation de la requête, en millisecondes.
statementCounters	`read`	Nombre de déclarations lues.
	`written`	Nombre de déclarations écrites.
	`deleted`	Nombre de déclarations supprimées.
transactionCounters	`committed`	Nombre de transactions effectuées.
	`rolledBack`	Nombre de transactions annulées.
vertexCounters	`added`	Nombre de sommets ajoutés.
	`removed`	Nombre de sommets supprimés.
	`propertiesAdded`	Nombre de propriétés de sommet ajoutées.

Groupe	Attribut	Description
	`propertiesRemoved`	Nombre de propriétés de sommet supprimées.
edgeCounters	`added`	Nombre d'arêtes ajoutées.
	`removed`	Nombre d'arêtes supprimées.
	`propertiesAdded`	Nombre de propriétés d'arête ajoutées.
	`propertiesRemoved`	Nombre de propriétés d'arête supprimées.
resultCache	`hitCount`	Nombre d'accès au cache de résultats.
	`missCount`	Nombre d'échec d'accès au cache de résultats.
	`putCount`	Nombre de mises en cache des résultats.
simultané ExecutionStats	`acceptedQueryCountAtStart`	Requêtes parallèles acceptées avec l'exécution de la requête actuelle au début.
	`runningQueryCountAtStart`	Requêtes parallèles exécutées avec l'exécution de la requête actuelle au début.
	`acceptedQueryCountAtEnd`	Requêtes parallèles acceptées avec l'exécution de la requête actuelle à la fin.
	`runningQueryCountAtEnd`	Requêtes parallèles exécutées avec l'exécution de la requête actuelle à la fin.

Groupe	Attribut	Description
queryBatch	`queryProcessingBatchSize`	Taille du lot lors du traitement des requêtes.
	`querySerialisationBatchSize`	Taille du lot lors de la sérialisation des requêtes.

Attributs de requête enregistrés en mode **de débogage**

Lorsque le paramètre du cluster de bases de données `neptune_enable_slow_query_log` est défini sur `debug`, les attributs suivants du compteur de stockage sont journalisés en plus des attributs journalisés en mode `info` :

Attribut	Description
`statementsScannedInAllIndexes`	Déclarations analysées dans tous les index.
`statementsScannedSPOGIndex`	Déclarations analysées dans l'index SPOG.
`statementsScannedPOGSIndex`	Déclarations analysées dans l'index POGS.
`statementsScannedGPSOIndex`	Déclarations analysées dans l'index GPSO.
`statementsScannedOSGPIndex`	Déclarations analysées dans l'index OSGP.
`statementsScannedInChunk`	Déclarations analysées ensemble par segments.
`postFilteredStatementScans`	Déclarations laissées après le filtrage post-analyse.
`distinctStatementScans`	Déclarations distinctes analysées.
`statementsReadInAllIndexes`	Déclarations lues après le filtrage post-analyse dans tous les index.
`statementsReadSPOGIndex`	Déclarations lues après le filtrage post-analyse dans l'index SPOG.

Attribut	Description
`statementsReadPOGSIndex`	Déclarations lues après le filtrage post-analyse dans l'index POGS.
`statementsReadGPSOIndex`	Déclarations lues après le filtrage post-analyse dans l'index GPSO.
`statementsReadOSGPIndex`	Déclarations lues après le filtrage post-analyse dans l'index OSGP.
`accessPathSearches`	Nombre de recherches de chemins d'accès.
`fullyBoundedAccessPathSearches`	Nombre de recherches de chemins d'accès par clé entièrement délimités.
`accessPathSearchedByPrefix`	Nombre de chemins d'accès recherchés par préfixe.
`searchesWhereRecordsWereFound`	Nombre de recherches ayant généré un ou plusieurs enregistrements en sortie.
`searchesWhereRecordsWereNotFound`	Nombre de recherches pour lesquelles aucun enregistrement n'a été généré.
`totalRecordsFoundInSearches`	Nombre total d'enregistrements trouvés à partir de toutes les recherches.
`statementsInsertedInAllIndexes`	Nombre de déclarations insérées dans tous les index.
`statementsUpdatedInAllIndexes`	Nombre de déclarations mises à jour dans tous les index.
`statementsDeletedInAllIndexes`	Nombre de déclarations supprimées dans tous les index.
`predicateCount`	Nombre de prédicats.

Attribut	Description
`dictionaryReadsFromValueToIdTable`	Nombre de lectures du dictionnaire de la table de valeurs à la table d'ID.
`dictionaryReadsFromIdToValueTable`	Nombre de lectures du dictionnaire à partir de l'ID de la table de valeurs.
`dictionaryWritesToValueToIdTable`	Nombre d'écritures du dictionnaire de la table de valeurs à la table d'ID.
`dictionaryWritesToIdToValueTable`	Nombre d'écritures du dictionnaire dans la table d'ID jusqu'à la table de valeurs.
`rangeCountsInAllIndexes`	Nombre de plages de valeurs dans tous les index.
`deadlockCount`	Nombre de blocages dans la requête.
`singleCardinalityInserts`	Nombre d'insertions de cardinalité uniques effectuées.
`singleCardinalityInsertDeletions`	Nombre de déclarations supprimées lors de l'insertion d'une cardinalité unique.
`sharedLocksWaitTimeMillis`	Nombre de millisecondes passées à attendre les verrous partagés.
`exclusiveLocksWaitTimeMillis`	Nombre de millisecondes passées à attendre des verrous exclusifs.

Exemple de journalisation du débogage pour une requête lente

L'exécution de la requête Gremlin suivante peut prendre plus de temps que le seuil défini pour les requêtes lentes :

```
gremlin=g.V().has('code','AUS').repeat(out().simplePath()).until(has('code','AGR')).path().by('
```

Si la journalisation des requêtes lentes était activée en mode de débogage, les attributs suivants seraient journalisés pour la requête, sous la forme suivante :


```
{
  "requestResponseMetadata": {
    "requestId": "5311e493-0e98-457e-9131-d250a2ce1e12",
    "requestType": "HTTP_GET",
    "responseStatusCode": 200
  },
  "queryStats": {
    "query":
 "gremlin=g.V().has('code','AUS').repeat(out().simplePath()).until(has('code','AGR')).path().by
    "queryFingerprint":
 "g.V().has(string0,string1).repeat(__.out().simplePath()).until(__.has(string0,string2)).path(
    "queryLanguage": "Gremlin"
  },
  "memoryStats": {
    "allocatedPermits": 20,
    "approximateUsedMemoryBytes": 14838
  },
  "queryTimeStats": {
    "startTime": "23/02/2023 11:42:52.657",
    "overallRunTimeMs": 2249,
    "executionTimeMs": 2229,
    "serializationTimeMs": 13
  },
  "statementCounters": {
    "read": 69979
  },
  "transactionCounters": {
    "committed": 1
  },
  "concurrentExecutionStats": {
    "acceptedQueryCountAtStart": 1
  },
  "queryBatchStats": {
    "queryProcessingBatchSize": 1000,
    "querySerialisationBatchSize": 1000
  },
  "storageCounters": {
    "statementsScannedInAllIndexes": 69979,
    "statementsScannedSPOGIndex": 44936,
    "statementsScannedPOGSIndex": 4,
```

```
    "statementsScannedGPSOIndex": 25039,
    "statementsReadInAllIndexes": 68566,
    "statementsReadSPOGIndex": 43544,
    "statementsReadPOGSIndex": 2,
    "statementsReadGPSOIndex": 25020,
    "accessPathSearches": 27,
    "fullyBoundedAccessPathSearches": 27,
    "dictionaryReadsFromValueToIdTable": 10,
    "dictionaryReadsFromIdToValueTable": 17,
    "rangeCountsInAllIndexes": 4,
    "sharedLocksWaitTimeMillis": 0,
    "exclusiveLocksWaitTimeMillis": 0
  }
}
```

Journalisation des appels d'API Amazon Neptune avec AWS CloudTrail

Amazon Neptune est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans Neptune. CloudTrail capture les appels d'API pour Neptune sous forme d'événements, y compris les appels depuis la console Neptune et les appels de code vers les API Neptune.

CloudTrail enregistre uniquement les événements pour les appels de l'API Neptune Management, tels que la création d'une instance ou d'un cluster. Si vous souhaitez auditer les modifications apportées à votre graphe, vous pouvez utiliser des journaux d'audit. Pour de plus amples informations, veuillez consulter Utilisation des journaux d'audit avec les cluster Amazon Neptune.

> **Important**
>
> Les appels à la console Amazon Neptune et à l'API sont enregistrés en tant qu'appels passés à l'API Amazon Relational Database Service (Amazon RDS). AWS CLI

Si vous créez un suivi, vous pouvez activer la diffusion continue d' CloudTrail événements vers un compartiment Amazon S3, y compris des événements pour Neptune. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez

déterminer la demande qui a été faite à Neptune, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des informations supplémentaires.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

Informations sur Neptune dans CloudTrail

CloudTrail est activé sur votre AWS compte lorsque vous le créez. Lorsqu'une activité a lieu dans Amazon Neptune, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez consulter, rechercher et télécharger les événements récents dans votre AWS compte. Pour plus d'informations, consultez la section Affichage des événements à l'aide de l'historique des CloudTrail événements.

Pour un enregistrement continu des événements de votre AWS compte, y compris des événements liés à Neptune, créez un parcours. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un journal de suivi dans la console, il s'applique à toutes les régions . Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence. Pour en savoir plus, consultez :

- Présentation de la création d'un journal d'activité
- CloudTrail Services et intégrations pris en charge
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers CloudTrail journaux de plusieurs régions et réception de fichiers CloudTrail journaux de plusieurs comptes

Si une action est entreprise au nom de votre AWS compte à l'aide de la console Neptune, de l'interface de ligne de commande Neptune ou des API du SDK Neptune, elle est enregistrée sous forme d'appels passés à AWS CloudTrail l'API Amazon RDS. Par exemple, si vous utilisez la console Neptune pour modifier une instance de base de données ou si vous appelez la commande AWS CLImodify-db-instance, le journal AWS CloudTrail indique un appel à l'action ModifyDBInstance de l'API Amazon RDS. Pour obtenir la liste des actions de l'API Neptune enregistrées par AWS CloudTrail, consultez le manuel Neptune API Reference.

> **Note**
>
> AWS CloudTrail enregistre uniquement les événements pour les appels de l'API Neptune Management, tels que la création d'une instance ou d'un cluster. Si vous souhaitez auditer les modifications apportées à votre graphe, vous pouvez utiliser des journaux d'audit. Pour de plus amples informations, veuillez consulter Utilisation des journaux d'audit avec les cluster Amazon Neptune.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été effectuée avec des informations d'identification d'utilisateur root ou IAM.
- Si la demande a été effectuée avec des informations d'identification de sécurité temporaires pour un rôle ou un utilisateur fédéré.
- Si la demande a été faite par un autre AWS service.

Pour plus d'informations, consultez la section Élément userIdentity CloudTrail .

Comprendre les entrées du fichier journal Neptune

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre un CloudTrail journal pour un utilisateur qui a créé un instantané d'une instance de base de données, puis l'a supprimée à l'aide de la console Neptune. La console est identifiée par l'élément `userAgent`. Les appels d'API requis effectués par la console (`CreateDBSnapshot` et `DeleteDBInstance`) se trouvent dans l'élément `eventName` pour chaque enregistrement. Il est possible de trouver des informations sur l'utilisateur (`Alice`) dans l'élément `userIdentity`.

```
{
  Records:[
```

```
  {
    "awsRegion":"us-west-2",
    "eventName":"CreateDBSnapshot",
    "eventSource":"domainSource",
    "eventTime":"2014-01-14T16:23:49Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-
kaos_fleet-1108-prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"0123456789012",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
    }
  },
  {
    "awsRegion":"us-west-2",
    "eventName":"DeleteDBInstance",
    "eventSource":"domainSource",
    "eventTime":"2014-01-14T16:28:27Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-
kaos_fleet-1108-prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"0123456789012",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
```

```
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
    }
  }
  ]
}
```

Événements Amazon Neptune

Rubriques

- Catégories et messages d'événements Amazon Neptune
- Afficher les événements Neptune
- Abonnement aux notifications d'événement Neptune
- Gestion des abonnements aux notifications d'événements Neptune

Amazon Neptune utilise Amazon Simple Notification Service (Amazon SNS) pour adresser des notifications lorsqu'un événement Neptune se produit. Ces notifications peuvent prendre n'importe quelle forme prise en charge par Amazon SNS pour une AWS région, comme un e-mail, un SMS ou un appel vers un point de terminaison HTTP.

Neptune regroupe ces événements en catégories auxquelles vous pouvez vous abonner afin que vous soyez informé lorsqu'un événement de cette catégorie se produit. Vous pouvez vous abonner à une catégorie d'événement pour une instance de base de données, un cluster de bases de données, un instantané de base de données, un instantané de cluster de bases de données ou un groupe de paramètres de base de données. Par exemple, si vous vous abonnez à la catégorie de sauvegarde d'une instance de base de données donnée, vous recevez une notification chaque fois que survient un événement lié à la sauvegarde et qui affecte l'instance de base de données. Vous recevez également une notification en cas de modification d'un abonnement à une notification d'événements.

Comme les événements se produisent au niveau du cluster et de l'instance, vous pouvez recevoir les événements si vous vous abonnez à un cluster de bases de données ou à une instance de base de données.

Les notifications d'événements sont envoyées aux adresses que vous fournissez lorsque vous créez l'abonnement. Il se peut que vous souhaitiez créer plusieurs abonnements différents, tel qu'un abonnement recevant toutes les notifications d'événements et un autre incluant uniquement les événements critiques affectant les instances de base de données de production. Vous pouvez facilement désactiver les notifications sans supprimer d'abonnement. Pour ce faire, réglez le bouton radio Activé sur Non dans la console Neptune.

> **Important**
>
> Amazon Neptune ne garantie pas l'ordre des événements envoyés dans un flux d'événements. L'ordre des événements est susceptible de changer.

Neptune utilise l'Amazon Resource Name (ARN) d'une rubrique Amazon SNS pour identifier chaque abonnement. La console Neptune crée l'ARN lorsque vous créez l'abonnement.

La facturation de la notification d'événement Neptune s'effectue via Amazon SNS. Des frais Amazon SNS s'appliquent en cas d'utilisation de la notification d'événement. Pour plus d'informations, consultez Tarification Amazon Simple Notification Service.

Catégories et messages d'événements Amazon Neptune

Neptune génère un nombre significatif d'événements dans les catégories auxquelles vous pouvez vous abonner à l'aide de la console Neptune. Chaque catégorie s'applique à un type de source, qui peut être une instance de base de données, un instantané de base de données ou un groupe de paramètres de base de données.

> **Note**
>
> Neptune utilise des définitions d'événements et des ID Amazon RDS existants.

Événements Neptune provenant d'instances de base de données

Le tableau suivant affiche une liste des événements par catégorie lorsqu'une instance de base de données est le type de source.

Catégorie	ID d'évènement Amazon RDS	Description
disponibilité	RDS-EVENT-0006	L'instance de base de données a redémarré.
	RDS-EVENT-0004	L’instance de base de données s’est arrêtée.
	RDS-EVENT-0022	Une erreur s'est produite lors du redémarrage du moteur Neptune.
sauvegarde	RDS-EVENT-0001	Sauvegarde de l’instance de base de données.
	RDS-EVENT-0002	Sauvegarde de l'instance de base de données terminée.
modification de configuration	RDS-EVENT-0009	L'instance de base de données a été ajoutée à un groupe de sécurité.
	RDS-EVENT-0024	L'instance de base de données est en cours de conversion en Multi-AZ instance de base de données.
	RDS-EVENT-0030	L'instance de base de données est en cours de conversion en Single-AZ instance de base de données.

Catégorie	ID d'évènement Amazon RDS	Description
	RDS-EVENT-0012	Application de la modification à la classe d'instance de base de données.
	RDS-EVENT-0018	Les paramètres de stockage de l'instanc e de base de données sont en cours de modification.
	RDS-EVENT-0011	Un groupe de paramètres de l'instance de base de données a changé.
	RDS-EVENT-0092	Un groupe de paramètres de l'instance de base de données a terminé sa mise à jour.
	RDS-EVENT-0028	Les sauvegardes automatiques de l'instance de base de données ont été désactivées.
	RDS-EVENT-0032	Les sauvegardes automatiques de l'instance de base de données ont été activées.

Catégorie	ID d'évènement Amazon RDS	Description
	RDS-EVENT-0025	L'instance de base de données a été convertie en Multi-AZ instance de base de données.
	RDS-EVENT-0029	L'instance de base de données a été convertie en Single-AZ instance de base de données.
	RDS-EVENT-0014	La classe d'instance de base de données de l'instance de base de données a été modifiée.
	RDS-EVENT-0017	Les paramètres de stockage de l'instanc e de base de données ont été modifiés.
	RDS-EVENT-0010	L'instance de base de données a été supprimée d'un groupe de sécurité.
création	RDS-EVENT-0005	Instance de base de données créée.
suppression	RDS-EVENT-0003	L'instance de base de données a été supprimée.

Catégorie	ID d'évènement Amazon RDS	Description
basculement	RDS-EVENT-0034	Neptune ne tente pas le basculement demandé, car un basculement s'est récemment produit sur l'instance de base de données.
	RDS-EVENT-0013	Un Multi-AZ basculement qui a entraîné la promotion d'une instance de secours a commencé.
	RDS-EVENT-0015	Un Multi-AZ basculement ayant entraîné la promotion d'une instance de secours est terminé. Le transfert du DNS vers la nouvelle instance de base de données principale peut prendre plusieurs minutes.
	RDS-EVENT-0065	L'instance a récupéré à partir d'un basculement partiel.
	RDS-EVENT-0049	Un Multi-AZ basculement est terminé.

Catégorie	ID d'évènement Amazon RDS	Description
	RDS-EVENT-0050	Une Multi-AZ activation a commencé après une restauration d'instance réussie.
	RDS-EVENT-0051	L' Multi-AZ activation est terminée. Votre base de données doit être accessible maintenant.
	RDS-EVENT-0031	L'instance de base de données a échoué en raison d'une configuration incompatible ou d'un problème de stockage sous-jacent. Commencez une restauration à un instant dans le passé pour l'instance de base de données.
	RDS-EVENT-0036	L'instance de base de données se trouve sur un réseau non compatible. Certains des ID de sous-réseau spécifiés ne sont pas valides ou n'existent pas.

Catégorie	ID d'évènement Amazon RDS	Description
	RDS-EVENT-0035	L'instance de base de données a des paramètres non valides. Par exemple, si l'instance de base de données n'a pas pu démarrer, un paramètre lié à la mémoire étant défini avec une valeur trop élevée pour cette classe d'instance, l'action du client consiste à modifier le paramètre et à redémarrer l'instance de base de données.
	RDS-EVENT-0082	Neptune n'a pas pu copier les données de sauvegarde d'un compartiment Amazon S3. Il est probable que les autorisations devant permettre à Neptune d'accéder au compartiment Amazon S3 soient mal configurées.

Catégorie	ID d'évènement Amazon RDS	Description
stockage faible	RDS-EVENT-0089	L'instance de base de données a consommé plus de 90 % de son stockage alloué. Vous pouvez surveiller l'espace de stockage pour une instance de base de données à l'aide de la métrique Free Storage Space (Espace de stockage libre).
	RDS-EVENT-0007	L'espace de stockage alloué pour l'instance de base de données a été épuisé. Pour résoudre le problème, vous devez allouer un stockage supplémentaire pour l'instance de base de données.
maintenance	RDS-EVENT-0026	La maintenance hors connexion de l'instance de base de données est en cours. L'instance de base de données n'est pas disponible actuellement.

Catégorie	ID d'évènement Amazon RDS	Description
	RDS-EVENT-0027	La maintenance hors connexion de l'instance de base de données est terminée. L'instance de base de données est désormais disponible.
	RDS-EVENT-0047	L'application des correctifs de l'instanc e de base de données s'est terminée.
notification	RDS-EVENT-0044	Operator-issued notification. Pour plus d'informations, consultez le message de l'événement.
	RDS-EVENT-0048	L'application des correctifs de l'instance de base de données a été retardée.
	RDS-EVENT-0087	L'instance de base de données a été arrêtée.
	RDS-EVENT-0088	L'instance de base de données a été démarrée.

Catégorie	ID d'évènement Amazon RDS	Description
	RDS-EVENT-0154	L'instance de base de données est en cours de démarrage dans la mesure où elle a dépassé le temps maximum autorisé pour son arrêt.
	RDS-EVENT-0158	L'instance de base de données est dans un état qui ne peut pas être mis à niveau.
	RDS-EVENT-0173	L'instance de base de données a été corrigée.
réplica en lecture	RDS-EVENT-0045	Une erreur s’est produite lors du processus de réplication en lecture. Pour plus d’informations, consultez le message de l’événement.

Catégorie	ID d'évènement Amazon RDS	Description
	RDS-EVENT-0046	Le réplica en lecture a repris la réplication. Ce message s'affiche lorsque vous créez un réplica en lecture, ou comme message de surveillance lorsque vous confirmez que la réplication fonctionn e correctement. Si ce message fait suite à une notification RDS-EVENT-0045 , la réplication a repris suite à une erreur ou à un arrêt de la réplication.
	RDS-EVENT-0057	La réplication sur le réplica en lecture est terminée.
	RDS-EVENT-0062	La réplication sur le réplica en lecture a été arrêtée manuellement.
	RDS-EVENT-0063	La réplication sur le réplica en lecture a été réinitialisée.

Catégorie	ID d'évènement Amazon RDS	Description
récupération	RDS-EVENT-0020	La récupération de l'instance de base de données a démarré. Le temps de récupération varie selon la quantité de données à restaurer.
	RDS-EVENT-0021	La récupération de l'instance de base de données est terminée.
	RDS-EVENT-0023	Une sauvegarde manuelle a été demandée, mais Neptune est en train de créer un instantané de base de données. Envoyez à nouveau la demande après que Neptune a terminé l'instantané de base de données.
	RDS-EVENT-0052	La restauration de l'Multi-AZ instance a commencé. Le temps de récupération varie selon la quantité de données à restaurer.
	RDS-EVENT-0053	La restauration de l'Multi-AZ instance est terminée.

Catégorie	ID d'évènement Amazon RDS	Description
restauration	RDS-EVENT-0008	L'instance de base de données a été restaurée à partir d'un instantané de base de données.
	RDS-EVENT-0019	L'instance de base de données a été restaurée à partir d'une sauvegarde à un instant dans le passé.

Événements Neptune provenant d'un cluster de bases de données

Le tableau suivant affiche une liste des événements par catégorie lorsqu'un cluster de bases de données est le type de source.

Catégorie	ID d'événement RDS	Description
basculement	RDS-EVENT-0069	Un basculement pour le cluster de base de données a échoué.
	RDS-EVENT-0070	Un basculement pour le cluster de base de données a redémarré.
	RDS-EVENT-0071	Un basculement pour le cluster de base de données s'est terminé.
	RDS-EVENT-0072	Un basculement pour le cluster de base de

Catégorie	ID d'événement RDS	Description
		données a commencé dans la même zone de disponibilité.
	RDS-EVENT-0073	Un basculement pour le cluster de base de données a commencé sur les zones de disponibilité.
	RDS-EVENT-0083	Neptune n'a pas pu copier les données de sauvegarde d'un compartiment Amazon S3. Il est probable que les autorisations devant permettre à Neptune d'accéder au compartiment Amazon S3 soient mal configurées.
maintenance	RDS-EVENT-0156	Une mise à niveau de version mineure du moteur de base de données est disponible pour le cluster de base de données.
notification	RDS-EVENT-0076	La migration vers un cluster de bases de données Neptune a échoué.

Catégorie	ID d'événement RDS	Description
	RDS-EVENT-0077	Une tentative de conversion d'une table de la base de données source en formulaire de base de données a échoué lors de la migration vers un cluster de bases de données Neptune.
	RDS-EVENT-0150	Le cluster de base de données s'est arrêté.
	RDS-EVENT-0151	Le cluster de base de données a démarré.
	RDS-EVENT-0152	L'arrêt du cluster de base de données a échoué.
	RDS-EVENT-0153	Le cluster de base de données est en cours de démarrage dans la mesure où il a dépassé le temps maximum autorisé pour son arrêt.

Événements Neptune provenant d'un instantané du cluster de bases de données

Le tableau suivant affiche la catégorie d'événement et la liste des événements lorsqu'un instantané de cluster de bases de données est le type de source.

Catégorie	ID d'événement RDS	Description
sauvegarde	RDS-EVENT-0074	La création d'un instantané de cluster de base de données manuel a commencé.
sauvegarde	RDS-EVENT-0075	Un instantané de cluster de base de données manuel a été créé.
notification	RDS-EVENT-0162	Échec de la tâche d'exportation de l'instantané de cluster de bases de données.
notification	RDS-EVENT-0163	La tâche d'exporta tion de l'instantané de cluster de bases de données a été annulée.
notification	RDS-EVENT-0164	La tâche d'exporta tion de l'instantané de cluster de bases de données est terminée.
sauvegarde	RDS-EVENT-0168	Création d'instantané de cluster automatisé.
sauvegarde	RDS-EVENT-0169	Instantané de cluster automatisé créé.
création	RDS-EVENT-0170	cluster de base de données créé.

Catégorie	ID d'événement RDS	Description
suppression	RDS-EVENT-0171	cluster de bases de données supprimé.
notification	RDS-EVENT-0172	Cluster de bases de données renommé de [ancien nom de cluster de bases de données] en [nouveau nom de cluster de bases de données].

Événements Neptune provenant du groupe de paramètres de cluster de bases de données

Le tableau suivant affiche la catégorie d'événement et la liste des événements lorsqu'un groupe de paramètres de cluster de bases de données est le type de source.

Catégorie	ID d'événement RDS	Description
modification de configuration	RDS-EVENT-0037	Le groupe de paramètres a été modifié.

Evénements Neptune en provenance d'un groupe de sécurité

Le tableau suivant affiche la catégorie d'événement et la liste des événements lorsqu'un groupe de sécurité de base de données est le type de source.

Catégorie	ID d'événement RDS	Description
modification de configuration	RDS-EVENT-0038	Le groupe de sécurité a été modifié.

Catégorie	ID d'événement RDS	Description
échec	RDS-EVENT-0039	Le groupe de sécurité dont [utilisateur] est propriétaire n'existe pas ; l'autorisation pour le groupe de sécurité a été révoquée.

Afficher les événements Neptune

Vous pouvez consulter les informations suivantes pour chaque événement Neptune :

- Nom de la ressource
- Type de ressource
- Catégorie d'événement
- Heure de l'événement
- Un message récapitulatif décrivant l'événement

Vous pouvez accéder aux événements dans les parties suivantes de la console AWS de gestion :

- La page Événements du volet de navigation de la console Neptune, qui affiche les événements des dernières 24 heures.
- Le tableau des événements récents de l'onglet Journaux et événements de la page détaillée d'un cluster de base de données ou d'une instance de base de données, qui peut afficher les événements survenus au cours des 14 derniers jours.

Vous pouvez également récupérer des événements à l'aide de la AWS CLI commande describe-events ou de l'opération DescribeEventsAPI. Avec l'API AWS CLI or, vous pouvez récupérer les événements survenus au cours des 14 derniers jours.

Affichage des événements (console)

Pour afficher les événements Neptune dans la console

1. Connectez-vous à la console de AWS gestion et ouvrez la console Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le panneau de navigation, sélectionnez Events.

   La page affiche les événements des dernières 24 heures. Vous pouvez filtrer les événements par type de source et par catégorie d'événements.

Affichage des événements (AWS CLI)

Pour afficher tous les événements Neptune générés au cours de la dernière heure, exécutez-le `describe-events` sans paramètres :

```
aws neptune describe-events
```

Pour afficher les événements des 7 derniers jours (10 080 minutes), utilisez le `--duration` paramètre :

```
aws neptune describe-events --duration 10080
```

Pour afficher les événements d'un cluster de base de données spécifique dans un intervalle de temps`--source-identifier`, utilisez les `--end-time` paramètres `--source-type``--start-time`,, et :

```
aws neptune describe-events \
    --source-identifier my-cluster \
    --source-type db-cluster \
    --start-time 2026-04-01T00:00Z \
    --end-time 2026-04-01T23:59Z
```

Pour plus d'informations, voir describe-events dans le manuel de référence des AWS CLI commandes. Pour récupérer des événements par programmation, reportez-vous DescribeEventsà la référence de l'API Neptune.

Abonnement aux notifications d'événement Neptune

Vous pouvez utiliser la console Neptune pour vous abonner aux notifications d'événements, comme suit :

Pour s'abonner à la notification d'événements Neptune

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le panneau de navigation, choisissez Abonnements aux événements.
3. Dans le volet Abonnements aux événements, choisissez Créer un abonnement aux événements.
4. Dans la boîte de dialogue Créer un abonnement aux événements, exécutez l'une des actions suivantes :

    a. Dans le champ Nom, entrez un nom pour l'abonnement à la notification d'événements.

    b. Pour Envoyez des notifications à, choisissez un ARN Amazon SNS existant pour une rubrique Amazon SNS ou choisissez créer une rubrique pour entrer le nom d'une rubrique et une liste de destinataires.

    c. Pour Type de source, choisissez un type de source.

    d. Choisissez Oui pour activer l'abonnement. Si vous souhaitez créer l'abonnement mais qu'aucune notification ne soit envoyée pour l'instant, choisissez Non.

    e. Selon le type de source que vous avez sélectionné, choisissez les catégories d'événement et les sources pour lesquelles vous souhaitez recevoir des notifications d'événements.

    f. Choisissez Créer.

Gestion des abonnements aux notifications d'événements Neptune

Si vous choisissez Abonnements aux événements dans le volet de navigation de la console Neptune, vous pouvez consulter les catégories d'abonnement et la liste de vos abonnements actuels.

Vous pouvez également modifier ou supprimer un abonnement spécifique.

Modification des abonnements aux notifications d'événement Neptune

Pour modifier vos abonnements aux notifications d'événement Neptune

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le panneau de navigation, choisissez Abonnements aux événements. Le volet Abonnements aux événements affiche tous les abonnements aux notifications d'événements.
3. Dans le volet Abonnements aux événements, choisissez l'abonnement que vous voulez modifier, puis choisissez Modifier.
4. Apportez les modifications requises à l'abonnement dans la section Cible ou Source. Vous pouvez ajouter ou supprimer des identifiants sources en les sélectionnant ou en annulant leur sélection dans la section Source.
5. Choisissez Modifier. La console Neptune indique que l'abonnement est en cours de modification.

Suppression d'un abonnement aux notifications d'événement Neptune

Vous pouvez supprimer un abonnement lorsque vous n'en avez plus besoin. Tous les abonnés à la rubrique ne reçoivent plus les notifications d'événements spécifiées par l'abonnement.

Pour supprimer un abonnement aux notifications d'événement Neptune

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le panneau de navigation, choisissez Abonnements aux événements.
3. Dans le volet Mes abonnements aux événements de base de données, cliquez sur l'abonnement que vous souhaitez supprimer.
4. Sélectionnez Delete (Supprimer).
5. La console Neptune indique que l'abonnement est en cours de suppression.

Marquage des ressources Amazon Neptune

Vous pouvez utiliser des balises Neptune pour ajouter des métadonnées à vos ressources Neptune. En outre, vous pouvez utiliser des balises associées à des politiques Gestion des identités et des accès AWS (IAM) pour gérer l'accès aux ressources Neptune et contrôler les actions qui peuvent

être appliquées à ces ressources. Enfin, vous pouvez utiliser des balises pour suivre les coûts en regroupant les dépenses pour des ressources balisées de la même façon.

Toutes les ressources Neptune peuvent être balisées, notamment les suivantes :

- Instances DB
- Clusters DB
- Réplicas en lecture
- Instantanés de base de données
- Instantanés de cluster DB
- Abonnements aux événements
- Groupes de paramètres DB
- Groupes de paramètres de cluster DB
- Groupes de sous-réseaux DB

Vue d'ensemble des balises de ressources Neptune

Une balise Amazon Neptune est une paire nom-valeur que vous définissez et associez à une ressource Neptune. Le nom s'appelle la clé. Fournir une valeur pour la clé est facultatif. Vous pouvez utiliser des balises pour affecter des informations arbitraires à une ressource Neptune. Vous pouvez utiliser une clé de balise, par exemple, pour définir une catégorie, et la valeur de balise peut être un élément de cette catégorie. Par exemple, vous pouvez définir une clé de balise appelée « projet » et une valeur de balise appelée « Salix », ce qui indique que la ressource Neptune est affectée au projet Salix. Vous pouvez également utiliser des balises pour désigner des ressources Neptune destinées aux tests ou à la production en utilisant une clé telle que `environment=test` ou `environment=production`. Nous vous recommandons d'utiliser un ensemble cohérent de clés de balise pour faciliter le suivi des métadonnées qui sont associées aux ressources Neptune.

Utilisez des balises pour organiser votre AWS facture afin de refléter votre propre structure de coûts. Pour ce faire, inscrivez-vous pour recevoir votre Compte AWS facture avec les valeurs clés du tag incluses. Ensuite, pour voir le coût de vos ressources combinées, organisez vos informations de facturation en fonction des ressources possédant les mêmes valeurs de clé de balise. Par exemple, vous pouvez baliser plusieurs ressources avec un nom d'application spécifique, puis organiser vos informations de facturation pour afficher le coût total de cette application dans plusieurs services.

Pour plus d'informations, consultez Utilisation des balises d'allocation des coûts dans le Guide de l'utilisateur AWS Billing .

Chaque ressource Neptune possède un ensemble de balises, qui contient toutes les balises qui sont attribuées à cette ressource Neptune. Un ensemble de balises peut contenir jusqu'à dix balises ou n'en contenir aucune. Si vous ajoutez une balise à une ressource Neptune ayant la même clé qu'une balise existante au niveau de cette ressource, la nouvelle valeur remplace l'ancienne.

AWS n'applique aucune signification sémantique à vos balises ; les balises sont interprétées strictement comme des chaînes de caractères. Neptune peut configurer des balises sur une instance de base de données ou d'autres ressources Neptune, en fonction des paramètres que vous utilisez lors de la création de la ressource. Par exemple, Neptune peut ajouter une balise indiquant qu'une instance de base de données est destinée à la production ou aux tests.

- La clé de balise correspond au nom obligatoire de la balise. La valeur de la chaîne peut comporter de 1 à 128 caractères Unicode et elle ne peut pas être précédée de « `aws:` » ou de « `rds:` ». La chaîne peut uniquement contenir l'ensemble de lettres, de chiffres et d'espaces Unicode, « _ », « . », « / », « = », « + », « - » (regex Java : « `^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$` »).
- La valeur de balise correspond à la valeur de chaîne facultative de la balise. La valeur de la chaîne peut comporter de 1 à 256 caractères Unicode et elle ne peut pas être précédée de « `aws:` ». La chaîne peut uniquement contenir l'ensemble de lettres, de chiffres et d'espaces Unicode, « _ », « . », « / », « = », « + », « - » (regex Java : « `^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$` »).

  Les valeurs comprises dans un ensemble de balises ne doivent pas nécessairement être uniques et peuvent être null. Par exemple, vous pouvez avoir une paire clé-valeur dans un ensemble de balises `project/Trinity` et `cost-center/Trinity`.

> **Note**
>
> Vous pouvez ajouter une balise à un instantané. Toutefois, votre facture ne reflètera pas ce groupement.

Vous pouvez utiliser l' AWS Management Console API Neptune ou l'API Neptune pour ajouter, répertorier et supprimer des balises sur les ressources Neptune. AWS CLI Lorsque vous utilisez l'API Neptune AWS CLI ou l'API Neptune, vous devez fournir le nom de ressource Amazon (ARN) de la ressource Neptune avec laquelle vous souhaitez travailler. Pour plus d'informations sur la création d'un ARN, consultez Création d'un ARN pour Neptune.

Les balises sont mises en cache à des fins d'autorisation. Pour cette raison, les ajouts et les mises à jour de balises sur les ressources Neptune peuvent prendre plusieurs minutes avant d'être disponibles.

Copier des tags dans Neptune

Lorsque vous créez ou restaurez une instance de base de données, vous pouvez spécifier que les balises de l'instance de base de données soient copiées vers les snapshots de l'instance de base de données. La copie des balises garantit que les métadonnées des instantanés de base de données correspondent à celles de l'instance de base de données source et que toutes les stratégies d'accès de l'instantané de base de données correspondent également à celles de l'instance de base de données source. Les balises ne sont pas copiées par défaut.

Vous pouvez spécifier que les balises soient copiées vers des snapshots DB pour les actions suivantes :

- Création d'une instance de base de données.
- Restauration d'une instance de base de données.
- Création d'un réplica en lecture.
- Copie d'un instantané de base de données.

> **Note**
>
> Si vous incluez une valeur pour le `--tag-key` paramètre de la AWS CLI commande create-db-cluster-snapshot (ou si vous fournissez au moins une balise à l'Créer un DBCluster instantanéaction d'API), Neptune ne copie pas les balises de l'instance de base de données source vers le nouvel instantané de base de données. C'est le cas même si l'option `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`) est activée sur l'instance de base de données source.
> Cela signifie que vous pouvez créer une copie d'une instance de base de données depuis un instantané de base de données sans avoir à ajouter de balises qui ne s'appliquent pas à la nouvelle instance de base de données. Après avoir créé votre instantané de base de données à l'aide de la AWS CLI `create-db-cluster-snapshot` commande (ou de l'action de l'API `CreateDBClusterSnapshot` Neptune), vous pouvez ajouter des balises comme décrit plus loin dans cette rubrique.

Marquage dans Neptune à l'aide du AWS Management Console

La processus de balisage d'une ressource Amazon Neptune est semblable pour toutes les ressources. La procédure suivante indique comment baliser une instance de base de données Neptune.

Pour ajouter une balise à une instance de base de données

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le panneau de navigation, choisissez Instances.

   > **Note**
   >
   > Pour filtrer la liste des instances de base de données dans le volet Instances, saisissez une chaîne de texte dans la zone Filtrer les instances. Seules les instances de base de données qui contiennent la chaîne apparaissent.

3. Choisissez l'instance de base de données que vous voulez baliser.
4. Choisissez Actions d'instance, puis Voir les informations
5. Dans la section des détails, faites défiler jusqu'à la section Balises.
6. Choisissez Ajouter. La fenêtre Ajouter des balises s'affiche.
7. Saisissez une valeur pour Clé de balise et Valeur.
8. Pour ajouter une autre balise, vous pouvez choisir Ajouter une autre balise et saisir une valeur pour Clé de balise et Valeur.

   Répétez cette étape autant de fois que nécessaire.
9. Choisissez Ajouter.

Pour supprimer une balise d'une instance de base de données

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le panneau de navigation, choisissez Instances.

> **Note**
>
> Pour filtrer la liste des instances de base de données dans le volet Instances, saisissez une chaîne de texte dans la zone Filtrer les instances. Seules les instances de base de données qui contiennent la chaîne apparaissent.

3. Choisissez l'instance de base de données que vous voulez baliser.
4. Choisissez Actions d'instance, puis Voir les informations
5. Dans la section des détails, faites défiler jusqu'à la section Balises.
6. Choisissez la balise que vous souhaitez supprimer.
7. Choisissez Supprimer, et choisissez Supprimer dans la fenêtre Supprimer des balises.

Marquage dans Neptune à l'aide du AWS CLI

Vous pouvez ajouter, répertorier ou supprimer des balises pour une instance de base de données dans Neptune à l'aide de l' AWS CLI.

- Pour ajouter une ou plusieurs balises à une ressource Neptune, utilisez la AWS CLI commande. `add-tags-to-resource`
- Pour répertorier les balises d'une ressource Neptune, utilisez la AWS CLI commande. `list-tags-for-resource`
- Pour supprimer une ou plusieurs balises d'une ressource Neptune, utilisez la AWS CLI commande. `remove-tags-from-resource`

Pour en savoir plus sur la création de l'Amazon Resource Name (ARN) requis, consultez Création d'un ARN pour Neptune.

Marquage dans Neptune à l'aide de l'API

Vous pouvez ajouter, répertorier ou supprimer des balises pour une instance de base de données à l'aide de l'API Neptune.

- Pour ajouter une balise à une ressource Neptune, utilisez l'opération `AddTagsToResource`.
- Pour répertorier les balises affectées à une ressource Neptune, utilisez l'opération `ListTagsForResource`.

- Pour supprimer des balises d'une ressource Neptune, utilisez l'opération `RemoveTagsFromResource`.

Pour en savoir sur la création de l'ARN requis, consultez Création d'un ARN pour Neptune.

Lorsque vous travaillez avec XML à l'aide de l'API Neptune, les balises utilisent le schéma suivant :

```
<Tagging>
    <TagSet>
        <Tag>
            <Key>Project</Key>
            <Value>Trinity</Value>
        </Tag>
        <Tag>
            <Key>User</Key>
            <Value>Jones</Value>
        </Tag>
    </TagSet>
</Tagging>
```

Le tableau suivant fournit une liste des balises XML autorisées et leurs caractéristiques. Les valeurs de `Key` et `Value` sont sensibles à la casse. Par exemple, `project=Trinity` et `PROJECT=Trinity` sont deux balises différentes.

Élément de balisage	Description
`TagSet`	Un ensemble de balises contient toutes les balises qui sont affectées à une ressource Neptune. Il ne peut y avoir qu'un ensemble de balises par ressource. Vous travaillez avec un ensemble de balises `TagSet` uniquement via l'API Neptune.
`Tag`	Une balise est une paire clé-valeur définie par l'utilisateur. Il peut y avoir de 1 à 50 balises dans un ensemble de balises.
`Key`	Une clé est le nom obligatoire de la balise. La valeur de la chaîne peut comporter de 1 à 128 caractères Unicode et elle ne peut pas être précédée de « `rds:` » ou de « `aws:` ». La chaîne peut uniquement contenir l'ensemble de lettres, de chiffres et d'espaces Unicode, « _ »,

Élément de balisage	Description
	« . », « / », « = », « + », « - » (regex Java : « `^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$` »). Les clés doivent être propres à un ensemble de balises. Par exemple, vous ne pouvez pas avoir une paire de clés dans un ensemble de balises avec la même clé mais des valeurs différentes, comme `project/Trinity` et `project/Xanadu` .
Value	Une valeur est la valeur facultative de la balise. La valeur de la chaîne peut comporter de 1 à 256 caractères Unicode et elle ne peut pas être précédée de « `rds:` » ou de « `aws:` ». La chaîne peut uniquement contenir l'ensemble de lettres, de chiffres et d'espaces Unicode, « _ », « . », « / », « = », « + », « - » (regex Java : « `^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$` »). Les valeurs comprises dans un ensemble de balises ne doivent pas nécessairement être uniques et peuvent être null. Par exemple, vous pouvez avoir une paire clé-valeur dans un ensemble de balises `project/Trinity` et `cost-center/Trinity` .

Utilisation des ARN administratifs dans Amazon Neptune

Les ressources créées dans Amazon Web Services sont chacune identifiées de façon unique par un Amazon Resource Name (ARN). Pour certaines opérations Amazon Neptune, vous devez identifier une ressource Neptune de façon unique en spécifiant son ARN.

> **Important**
>
> Amazon Neptune partage le format des ARN Amazon RDS pour les actions administratives utilisant l'Référence de l'API de gestion. Les ARN administratifs Neptune contiennent `rds` et non `neptune-db`. Pour les ARN du plan de données qui identifient les ressources de données Neptune, consultez Spécification des ressources de données.

Rubriques

- Création d'un ARN pour Neptune

- Obtenir un ARN existant dans Amazon Neptune

Création d'un ARN pour Neptune

Vous pouvez créer un ARN pour une ressource Amazon Neptune en utilisant la syntaxe suivante. Notez que Neptune partage le format des ARN Amazon RDS.

`arn:aws:rds:<region>:<account number>:<resourcetype>:<name>`

Le tableau suivant indique le format à utiliser lors de la création d'un ARN pour un type de ressource administative Neptune spécifique.

Type de ressource	Format ARN
instance de base de données	arn:aws:rds : : *<region>* *<account>* :db: *<name>* Par exemple : `arn:aws:rds: us-east-2 :123456789012 :db:my-instance-1`
Cluster DB	arn:aws:rds : : *<region>* *<account>* :cluster: *<name>* Par exemple : `arn:aws:rds: us-east-2 :123456789012 :cluster: my-cluster-1`
Abonnement aux événements	arn:aws:rds : : *<region>* *<account>* :es: *<name>* Par exemple : `arn:aws:rds: us-east-2 :123456789012 :es:my-subscription`
Groupe de paramètres de base de données	arn:aws:rds : : *<region>* *<account>* :pg: *<name>* Par exemple :

Type de ressource	Format ARN
	`arn:aws:rds: us-east-2 :123456789012 :pg:my-param-enable-logs`
Groupe de paramètres de cluster de bases de données	arn:aws:rds : : *<region>* *<account>* :cluster-pg: *<name>* Par exemple : `arn:aws:rds: us-east-2 :123456789012 :cluster-pg: my-cluster-param-timezone`
Instantané de cluster DB	fil : aws : rds: *<region>* *<account>* :cluster-snapshot: *<name>* Par exemple : `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-snap-20160809`
Groupe de sous-réseaux de base de données	fil : aws : rds: *<region>* *<account>* :subgrp: *<name>* Par exemple : `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet-10`

Obtenir un ARN existant dans Amazon Neptune

Vous pouvez obtenir l'ARN d'une ressource Neptune en utilisant l'API AWS Management Console, AWS Command Line Interface (AWS CLI) ou Neptune.

Obtenir un ARN existant à l'aide du AWS Management Console

Pour obtenir un ARN à l'aide de la console, accédez à la ressource pour laquelle vous souhaitez obtenir un ARN, puis consultez les détails de cette ressource. Par exemple, pour obtenir l'ARN d'une instance de base de données, choisissez Instances dans le volet de navigation, puis choisissez

l'instance souhaitée dans la liste. L'ARN se trouve dans la section Instance Details (Détails de l'instance).

Obtenir un ARN existant à l'aide du AWS CLI

Pour utiliser le AWS CLI afin d'obtenir un ARN pour une ressource Neptune particulière, utilisez la `describe` commande correspondant à cette ressource. Le tableau suivant indique chaque AWS CLI commande et la propriété ARN utilisée avec la commande pour obtenir un ARN.

AWS CLI Commande	Propriété d'ARN
describe-event-subscriptions	EventSubscriptionArn
describe-certificates	CertificateArn
describe-db-parameter-groups	DBParameterGroupArn
describe-db-cluster-parameter-groups	DBClusterParameterGroupArn
describe-db-instances	DBInstanceArn
describe-events	SourceArn
describe-db-subnet-groups	DBSubnetGroupArn
describe-db-clusters	DBClusterArn
describe-db-cluster-snapshots	DBClusterSnapshotArn

Par exemple, la AWS CLI commande suivante obtient l'ARN d'une instance de base de données.

Example

Pour Linux, OS X ou Unix :

```
aws neptune describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2
```

Pour Windows :

```
aws neptune describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2
```

Obtenir un ARN existant à l'aide de l'API

Pour obtenir un ARN pour une ressource Neptune particulière, appelez les actions d'API suivantes et utilisez les propriétés d'ARN indiquées.

Action d'API Neptune	Propriété d'ARN
DescribeEventSubscriptions	EventSubscriptionArn
DescribeCertificates	CertificateArn
DescribeDBParameterGroups	DBParameterGroupArn
DescribeDBClusterParameterGroups	DBClusterParameterGroupArn
DescribeDBInstances	DBInstanceArn
DescribeEvents	SourceArn
DescribeDBSubnetGroups	DBSubnetGroupArn
DescribeDBClusters	DBClusterArn
DescribeDBClusterSnapshots	DBClusterSnapshotArn

Sauvegarde et restauration d'un cluster de bases de données Amazon Neptune

Cette section explique comment sauvegarder et restaurer des clusters de bases de données Amazon Neptune.

Rubriques

- Sauvegardes Neptune
- Tolérance aux pannes pour un cluster de bases de données Neptune
- CloudWatch métriques utiles pour gérer le stockage de sauvegarde Neptune
- Restauration des données à partir d'une sauvegarde Neptune
- Fenêtre de sauvegarde dans Neptune
- Création d'un instantané de cluster de bases de données dans Neptune
- Restauration à partir d'un instantané de cluster de base de données
- Copie d'un instantané de cluster de bases de données
- Partage d'un instantané de cluster de base de données
- Suppression d'un instantané Neptune

Sauvegardes Neptune

Neptune sauvegarde automatiquement le volume du cluster et conserve les données de restauration pendant la totalité de la période de rétention des sauvegardes. Les sauvegardes Neptune étant continues et incrémentielles, vous pouvez rapidement opérer une restauration à un point quelconque de la période de rétention des sauvegardes. Aucun impact sur les performances ou interruption du service de base de données ne se produit lors de l'écriture des données de sauvegarde. Vous pouvez spécifier une période de rétention des sauvegardes, comprise entre 1 et 35 jours, lorsque vous créez ou modifiez un cluster de base de données.

Pour contrôler votre utilisation du stockage des sauvegardes, vous pouvez réduire l'intervalle de rétention des sauvegardes, supprimer d'anciens instantanés manuels lorsqu'ils ne sont plus nécessaires, ou les deux. Pour faciliter la gestion des coûts, vous pouvez surveiller la quantité de stockage consommée par les sauvegardes continues et les instantanés manuels qui persistent au-

delà de la période de rétention. Vous pouvez réduire l'intervalle de rétention des sauvegardes et supprimer les instantanés manuels lorsqu'ils ne sont plus nécessaires.

Si vous souhaitez conserver une sauvegarde au-delà de la période de rétention, vous pouvez aussi prendre un instantané des données dans votre volume de cluster. Le stockage des instantanés est soumis aux frais standard de stockage pour Neptune. Pour plus d'informations sur la tarification du stockage d'instance, consultez Tarification Amazon Neptune.

Neptune conserve les données des restaurations incrémentielles pendant la totalité de la période de rétention des sauvegardes. Par conséquent, vous avez uniquement besoin de créer un instantané pour les données que vous souhaitez garder au-delà de la période de rétention des sauvegardes. Vous pouvez créer un nouveau cluster de bases de données à partir de l'instantané.

Important

Si vous supprimez un cluster de bases de données, toutes ses sauvegardes automatiques sont supprimées en même temps et ne peuvent pas être récupérées. En d'autres termes, si vous ne choisissez pas de créer manuellement un instantané de bases de données final, vous ne pourrez pas restaurer l'instance de base de données à son état final ultérieurement. Les instantanés manuels ne sont pas supprimés lorsque vous supprimez un cluster.

Note

- Pour les clusters de bases de données Amazon Neptune, la période de rétention des sauvegardes par défaut est d'une journée quel que soit le mode de création du cluster de bases de données.
- Vous ne pouvez pas désactiver les sauvegardes automatiques sur Neptune. La période de rétention des sauvegardes pour Neptune est gérée par le cluster de bases de données.

Tolérance aux pannes pour un cluster de bases de données Neptune

De par sa conception, un cluster de bases de données Neptune est tolérant aux pannes. Le volume de cluster couvre plusieurs zones de disponibilité dans une seule AWS région, et chaque zone de disponibilité contient une copie des données du volume de cluster. Cette fonctionnalité signifie que

votre cluster de bases de données peut tolérer une défaillance d'une zone de disponibilité sans perte de données et uniquement une brève interruption de service.

En cas de défaillance de l'instance principale d'un cluster de bases de données, Neptune bascule automatiquement vers une nouvelle instance principale de l'une des deux façons suivantes :

- Par la promotion d'un réplica Neptune existant vers la nouvelle instance principale
- En créant une autre instance principale

Si le cluster de bases de données possède un ou plusieurs réplicas Neptune, un réplica Neptune est promu vers l'instance principale lors d'un événement d'échec. Un événement d'échec se traduit par une brève interruption, pendant laquelle les opérations de lecture et d'écriture échouent avec une exception. Cependant, le service est généralement restauré en moins de 120 secondes, et souvent en moins de 60 secondes. Pour augmenter la disponibilité de votre cluster de bases de données, nous vous recommandons de créer au moins un réplica Neptune dans deux zones de disponibilité ou plus.

Vous pouvez personnaliser l'ordre dans lequel les réplicas Neptune sont promus vers l'instance principale après un échec, en affectant à chaque réplica une priorité. Les priorités s'étendent de la valeur 0 pour la plus haute priorité à la valeur 15 pour la plus basse priorité. Si l'instance principale échoue, Neptune promeut le réplica Neptune avec la plus haute priorité vers la nouvelle instance principale. Vous pouvez modifier la priorité d'un réplica Neptune à tout moment. La modification de la priorité ne déclenche pas un basculement.

Vous pouvez utiliser le AWS CLI pour définir la priorité de basculement d'une instance de base de données, comme suit :

```
aws neptune modify-db-instance --db-instance-identifier (the instance ID) --promotion-
tier (the failover priority value)
```

Plusieurs réplicas Neptune peuvent partager la même priorité, ce qui se traduit par des niveaux de promotion. Si deux réplicas Neptune ou plus partagent la même priorité, Neptune promeut le réplica le plus grand en taille. Si deux réplicas Neptune ou plus partagent les mêmes priorité et taille, Neptune promeut un réplica arbitraire du même niveau de promotion.

Si le cluster de bases de données ne contient aucun réplica Neptune, l'instance principale est recréée pendant un événement d'échec. Un événement d'échec se traduit par une interruption, pendant laquelle les opérations de lecture et d'écriture échouent avec une exception. Le service est rétabli

quand la nouvelle instance principale est créée, ce qui prend généralement moins de 10 minutes. La promotion d'un réplica Neptune vers l'instance principale est beaucoup plus rapide que la création d'une instance principale.

CloudWatch métriques utiles pour gérer le stockage de sauvegarde Neptune

Vous pouvez utiliser les CloudWatch métriques Amazon `TotalBackupStorageBilled`, `SnapshotStorageUsed`, et `BackupRetentionPeriodStorageUsed` pour examiner et surveiller la quantité de stockage utilisée par vos sauvegardes Neptune, comme suit :

- `BackupRetentionPeriodStorageUsed` représente la quantité de stockage des sauvegardes utilisée, en octets, pour stocker les sauvegardes continues à l'heure actuelle. Cette valeur dépend de la taille du volume de cluster et de la quantité de modifications apportées au cours de la période de rétention. Cependant, à des fins de facturation, elle ne dépasse pas la taille du volume de cluster cumulée au cours de la période de rétention. Par exemple, si la taille `VolumeBytesUsed` de votre cluster est de 107 374 182 400 octets (100 Gio) et que votre période de conservation est de deux jours, la valeur maximale `BackupRetentionPeriodStorageUsed` est de 214 748 364 800 octets (100 Gio +100 Gio).
- `SnapshotStorageUsed` représente la quantité de stockage des sauvegardes utilisée, en octets, pour stocker les instantanés manuels au-delà de la période de conservation des sauvegardes. Les instantanés manuels ne portent pas préjudice au stockage des sauvegardes d'instantanés tant que l'horodatage de leur création se trouve dans la période de conservation. Il en va de même pour les sauvegardes d'instantanés automatiques. La taille de chaque instantané correspond à la taille du volume de cluster au moment où vous avez pris l'instantané. La valeur de `SnapshotStorageUsed` dépend du nombre d'instantanés que vous conservez et de la taille de chaque instantané. Par exemple, supposons que vous disposez d'un instantané manuel en dehors de la période de conservation et que la taille `VolumeBytesUsed` du cluster était de 100 Gio lorsque l'instantané a été pris. La quantité de SnapshotStorageUsed est de 107 374 182 400 octets (100 GiB).
- `TotalBackupStorageBilled` représente la somme de `BackupRetentionPeriodStorageUsed` et `SnapshotStorageUsed`, en octets, moins un volume de stockage de sauvegarde disponible égal à la taille du volume de cluster pour une journée. Le stockage de sauvegarde disponible est égal à la taille de volume la plus récente. Par exemple, si la taille `VolumeBytesUsed` de votre cluster est de 100 Gio, votre période de

conservation est de deux jours. Si vous disposez d'un instantané manuel en dehors de la période de conservation, `TotalBackupStorageBilled` est de 214 748 364 800 octets (200 Gio + 100 Gio + 100 Gio).

Vous pouvez surveiller un cluster Neptune et créer des rapports à l'aide de CloudWatch métriques via la CloudWatch console. Pour plus d'informations sur l'utilisation CloudWatch des métriques, consultez Surveillance de Neptune et le tableau des métriques dansMétriques Neptune CloudWatch.

Restauration des données à partir d'une sauvegarde Neptune

Vous pouvez récupérer vos données en créant un cluster de bases de données Neptune à partir des données de sauvegarde que Neptune conserve ou d'un instantané de cluster de bases de données que vous avez enregistré. Vous pouvez restaurer rapidement une nouvelle copie d'un cluster de bases de données créé à partir des données de sauvegarde à un point quelconque de la période de rétention des sauvegardes. La nature continue et incrémentielle des sauvegardes Neptune pendant la période de rétention signifie que vous n'avez pas besoin d'effectuer fréquemment des instantanés de vos données pour améliorer les temps de restauration.

Pour déterminer la date de restauration possible la plus ancienne ou la plus récente d'une instance de base de données, recherchez les valeurs `Latest Restorable Time` ou `Earliest Restorable Time` dans la console Neptune. La dernière date de restauration d'un cluster de base de données est le point le plus récent auquel vous pouvez restaurer votre cluster de base de données, généralement dans les 5 minutes qui précèdent l'heure actuelle. L'heure de restauration la plus ancienne spécifie jusqu'à quelle date vous pouvez remonter au sein de la période de rétention des sauvegardes pour restaurer votre volume de cluster.

Vous pouvez déterminer à quel moment la restauration d'un cluster de bases de données est complète à l'aide des valeurs `Latest Restorable Time` et `Earliest Restorable Time`. Les valeurs `Latest Restorable Time` et `Earliest Restorable Time` retournent NULL tant que l'opération de restauration n'est pas terminée. Vous ne pouvez pas demander une opération de sauvegarde ou de restauration si les valeurs `Latest Restorable Time` ou `Earliest Restorable Time` retournent NULL.

Pour restaurer une instance de base de données à une heure spécifiée à l'aide du AWS Management Console

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.

2. Dans le panneau de navigation, choisissez Instances. Choisissez l'instance principale du cluster de bases de données que vous voulez restaurer.
3. Choisissez Instance actions (Actions d'instance), puis Restore to point in time (Restaurer à un moment donné).

   Dans la fenêtre Launch DB Instance (Lancer une instance de base de données), choisissez Custom (Personnalisé) sous Restore time (Heure de restauration).
4. Spécifiez la date et l'heure de la restauration souhaitée sous Custom (Personnalisé).
5. Attribuez un nom à la nouvelle instance de base de données restaurée dans DB instance identifier (Identifiant de l'instance de base de données) sous Settings (Paramètres).
6. Choisissez Launch DB Instance (Lancer une instance de base de données) pour lancer l'instance de base de données restaurée.

   Une nouvelle instance de base de données est créée avec le nom que vous avez spécifié et un nouveau cluster de base de données est créé. Le nom du cluster de base de données correspond au nom de la nouvelle instance de base de données suivi de `-cluster`. Par exemple, si le nom de la nouvelle instance de base de données est `myrestoreddb`, le nom du nouveau cluster de base de données est `myrestoreddb-cluster`.

Fenêtre de sauvegarde dans Neptune

Les sauvegardes automatiques sont exécutées chaque jour pendant la fenêtre de sauvegarde préférée. Si la sauvegarde a besoin de plus de temps que la durée allouée par la fenêtre de sauvegarde, elle continue après la fin de la fenêtre jusqu'à ce qu'elle soit terminée. La fenêtre de sauvegarde ne peut pas chevaucher la fenêtre de maintenance hebdomadaire pour l'instance de base de données.

Pendant la fenêtre de sauvegarde automatique, le stockage I/O peut être suspendu brièvement pendant l'initialisation du processus de sauvegarde (généralement en moins de quelques secondes). Vous pouvez rencontrer des latences élevées pendant quelques minutes lors de sauvegardes de déploiements multi-AZ.

La fenêtre de sauvegarde est normalement sélectionnée de manière aléatoire à partir d'un bloc de huit heures par région par le plan de contrôle Amazon RDS sur lequel repose Neptune. Les blocs de temps pour chaque région à partir de laquelle les fenêtres de sauvegarde par défaut sont affectées sont documentés dans la section relative à la fenêtre de sauvegarde dans le guide de l'utilisateur Amazon RDS.

Création d'un instantané de cluster de bases de données dans Neptune

Neptune crée un instantané du volume de stockage de votre cluster de bases de données en sauvegardant l'intégralité de ce dernier, et pas seulement les bases de données. Lorsque vous créez un instantané de cluster de base de données, vous devez identifier quel cluster de base de données vous allez sauvegarder. Ensuite, nommez votre instantané de cluster de base de données afin que vous puissiez restaurer ultérieurement à partir de ce dernier. Le temps nécessaire à la création d'un instantané de cluster DB varie en fonction de la taille de vos bases de données. L'instantané inclut l'intégralité du volume de stockage. Par conséquent, la taille des fichiers (tels que les fichiers temporaires) a également une incidence sur le temps nécessaire à la création de l'instantané.

Vous pouvez créer un instantané de cluster de base de données à l' AWS Management Console aide de l'API AWS CLI Neptune ou de l'API Neptune.

Utiliser la console pour créer un instantané de cluster de base de données

Pour créer un instantané de cluster de bases de données

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Dans la liste des instances DB, choisissez l'instance principale du cluster DB.
4. Choisissez Actions d'instance, puis Prendre un instantané.

   La fenêtre Capture d'un instantané DB apparaît.
5. Entrez le nom de l'instantané de cluster de base de données dans la zone Nom de l'instantané.
6. Choisissez Prendre un instantané.

Restauration à partir d'un instantané de cluster de base de données

Lorsque vous créez l'instantané Amazon Neptune d'un cluster de bases de données, Neptune génère un instantané du volume de stockage de ce cluster en sauvegardant l'intégralité de ses données, pas seulement les instances. Vous pourrez ultérieurement créer un cluster de bases de données en effectuant une restauration à partir de cet instantané. Lorsque vous restaurez le cluster de bases de données, indiquez le nom de l'instantané à partir duquel la restauration doit être effectuée, puis spécifiez le nom du cluster de bases de données qui sera créé par cette restauration.

Table des matières

- Points à prendre en compte lors de la restauration d'un cluster de bases de données Neptune à partir d'un instantané
  - Vous ne pouvez pas effectuer la restauration dans un cluster de bases de données préexistant
  - Aucune instance n'est restaurée.
  - Aucun groupe de paramètres personnalisés n'est restauré.
  - Aucun groupe de sécurité personnalisé n'est restauré.
  - Vous ne pouvez pas effectuer la restauration à partir d'un instantané partagé et chiffré.
  - Un cluster de bases de données restauré utilise autant d'espace de stockage qu'auparavant
- Comment effectuer une restauration à partir d'un instantané
  - Utiliser la console pour effectuer une restauration à partir d'un instantané

Points à prendre en compte lors de la restauration d'un cluster de bases de données Neptune à partir d'un instantané

Vous ne pouvez pas effectuer la restauration dans un cluster de bases de données préexistant

Le processus de restauration génère toujours un nouveau cluster de bases de données. La restauration ne peut donc pas avoir lieu dans un cluster de bases de données qui existe déjà.

Aucune instance n'est restaurée.

Aucune instance n'est associée à un nouveau cluster de bases de données créé par une restauration.

Dès que la restauration est terminée et que le nouveau cluster de bases de données est disponible, créez de manière explicite les instances dont vous avez besoin. Pour ce faire, vous pouvez utiliser la console Neptune ou l'API Créez DBInstance.

Aucun groupe de paramètres personnalisés n'est restauré.

Un nouveau cluster de bases de données créé par une restauration est automatiquement associé au groupe de paramètres de base de données par défaut.

Dès que la restauration est terminée et que le nouveau cluster de bases de données est disponible, vous devez associer les groupes de paramètres de base de données personnalisés utilisés par l'instance à partir de laquelle vous avez effectué la restauration. Pour ce faire, utilisez la commande Modify de la console Neptune ou de l'API Modifier DBInstance.

> **Important**
>
> Nous vous recommandons d'enregistrer un groupe de paramètres personnalisés utilisé dans le cluster de bases de données dont vous créez un instantané. Lorsque vous effectuerez une restauration à partir de cet instantané, vous pourrez facilement associer le groupe de paramètres approprié au cluster de bases de données restauré.

Aucun groupe de sécurité personnalisé n'est restauré.

Un groupe de sécurité par défaut est automatiquement associé au nouveau cluster de bases de données créé par une restauration.

Dès que la restauration est terminée et que votre nouveau cluster de bases de données est disponible, associez les groupes de sécurité personnalisés utilisés par l'instance à partir de laquelle vous avez effectué la restauration. Pour ce faire, utilisez la commande Modify de la console Neptune ou de l'API Modifier DBInstance.

Vous ne pouvez pas effectuer la restauration à partir d'un instantané partagé et chiffré.

Vous ne pouvez pas restaurer un cluster de bases de données à partir d'un instantané de cluster de bases de données qui est à la fois partagé et chiffré.

Au lieu de cela, créez une copie de l'instantané et restaurez le cluster à partir de cette copie.

Pour plus d'informations sur la gestion du chiffrement lors de la copie d'un instantané, consultezGestion du chiffrement lors de la copie d'instantané. Pour plus d'informations sur le partage d'instantanés chiffrés, consultezPartage d'un instantané de cluster de base de données chiffré.

Un cluster de bases de données restauré utilise autant d'espace de stockage qu'auparavant

Lorsque vous restaurez un cluster DB à partir d'un instantané de cluster DB, la quantité de stockage allouée au nouveau cluster est la même que celle allouée au cluster DB à partir duquel l'instantané a été créé, quelle que soit la quantité de stockage allouée réellement utilisée.

En d'autres termes, la limite supérieure pour laquelle vous êtes facturé ne change pas. Pour redéfinir la limite supérieure, vous devez exporter les données de votre graphique, puis les recharger sur un nouveau cluster DB (voir Facturation du stockage Neptune).

Comment effectuer une restauration à partir d'un instantané

Vous pouvez restaurer un cluster de base de données à partir d'un instantané de cluster de base de données à l' AWS Management Console aide de l' AWS CLI API Neptune ou de l'API Neptune.

Utiliser la console pour effectuer une restauration à partir d'un instantané

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le panneau de navigation, choisissez Snapshots.
3. Choisissez l'instantané de cluster de bases de données à partir duquel vous voulez restaurer.
4. Sélectionnez Actions, puisRestore Snapshot (Restaurer l'instantané).
5. Sur la page Restaurer l'instance DB, dans la zone Identificateur de l'instance DB, saisissez le nom de votre cluster DB restauré.
6. Choisissez Restore DB Instance.
7. Si vous voulez restaurer la fonctionnalité du cluster DB sur celle du cluster DB à partir duquel l'instantané a été créé, vous devez modifier le cluster DB afin qu'il utilise le groupe de sécurité. Les étapes suivantes supposent que votre cluster DB se trouve dans un VPC. Si votre cluster de bases de données ne se trouve pas dans un VPC, utilisez la console Amazon EC2 pour trouver le groupe de sécurité dont vous avez besoin pour ce cluster.

    a. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.

b. Dans le panneau de navigation, choisissez Groupes de sécurité.

c. Sélectionnez le groupe de sécurité que vous souhaitez utiliser pour vos clusters DB. Si nécessaire, ajoutez des règles pour associer le groupe de sécurité à un groupe de sécurité pour une instance EC2.

Copie d'un instantané de cluster de bases de données

Avec Neptune, vous pouvez copier des instantanés de cluster de bases de données automatisés ou manuels. Après avoir copié un instantané, la copie est un instantané manuel.

Vous pouvez copier un instantané au sein d'une même AWS région ou d'une région à AWS l'autre. Les instantanés peuvent être copiés entre les régions et les comptes en une seule étape.

La copie d'un instantané automatique vers un autre AWS compte s'effectue en deux étapes : vous créez d'abord un instantané manuel à partir de l'instantané automatique, puis vous copiez l'instantané manuel sur l'autre compte.

Au lieu de copier, vous pouvez également partager des instantanés manuels avec d'autres AWS comptes. Pour de plus amples informations, veuillez consulter Partage d'un instantané de cluster de base de données.

Rubriques

- Limitations de la copie d'un instantané
- Conservation des copies d'instantané de cluster de base de données
- Gestion du chiffrement lors de la copie d'instantané
- Copier des instantanés AWS Régions
- Copier un instantané de cluster de base de données à l'aide de la console
- Copier un instantané de cluster de base de données à l'aide du AWS CLI

Limitations de la copie d'un instantané

Vous trouverez ci-dessous certaines limites qui s'appliquent lorsque vous copiez des instantanés :

- Vous pouvez copier un instantané entre la Chine (Pékin) et la Chine (Ningxia), mais vous ne pouvez pas copier un instantané entre ces régions chinoises et d'autres AWS régions.
- Vous pouvez copier un instantané entre AWS GovCloud (US-East) et AWS GovCloud (US-West), mais vous ne pouvez pas copier un instantané entre ces AWS GovCloud (US) régions et d'autres AWS régions.
- Si vous supprimez un instantané source avant que l'instantané cible ne soit disponible, la copie d'instantané peut échouer. Vérifiez que l'instantané cible a le statut `AVAILABLE` avant de supprimer un instantané source.

- Vous pouvez avoir jusqu'à cinq demandes de copies d'instantanés en cours dans une seule région par compte.
- Selon les régions impliquées et le volume de données à copier, la copie d'un instantané entre régions peut prendre plusieurs heures.

  S'il existe un grand nombre de demandes de copie instantanée entre régions provenant d'une AWS région source donnée, Neptune peut placer les nouvelles demandes de copie entre régions provenant de cette région AWS source dans une file d'attente jusqu'à ce que certaines copies en cours soient terminées. Aucune information d'avancement n'est affichée sur les demandes de copie quand elles sont dans cette file d'attente. Les informations d'avancement ne sont affichées qu'après le démarrage de la copie.

Conservation des copies d'instantané de cluster de base de données

Neptune supprime automatiquement les instantanés comme suit :

- A la fin de leur période de conservation.
- Lorsque vous désactivez les instantanés automatiques pour un cluster de base de données.
- Lorsque vous supprimez un cluster de bases de données.

Si vous souhaitez conserver un instantané de base de données à plus long terme, copiez-le pour créer un instantané de base de données manuel qui sera conservé jusqu'à ce que vous le supprimiez. Des coûts de stockage Neptune peuvent s'appliquer aux instantanés manuels si ces derniers dépassent votre espace de stockage par défaut.

Pour plus d'informations sur les coûts de stockage des sauvegardes, consultez la section Tarification de Neptune.

Gestion du chiffrement lors de la copie d'instantané

Vous pouvez copier un instantané chiffré à l'aide d'une clé de AWS KMS chiffrement. Si vous copiez un instantané chiffré, la copie de l'instantané doit également être chiffrée. Vous pouvez chiffrer la copie avec la même clé de AWS KMS chiffrement que l'instantané d'origine, ou vous pouvez spécifier une autre clé de AWS KMS chiffrement.

Vous ne pouvez pas chiffrer un instantané de cluster de bases de données non chiffré lorsque vous le copiez.

Pour les instantanés de cluster de base de données Amazon Neptune, vous pouvez également laisser l'instantané de cluster de base de données non chiffré et spécifier une clé de AWS KMS chiffrement lors de la restauration. Le cluster de bases de données restauré est chiffré à l'aide de la clé indiquée.

Copier des instantanés AWS Régions

Lorsque vous copiez un instantané AWS dans une région différente de la AWS région de l'instantané source, la première copie est une copie complète, même si vous copiez un instantané incrémentiel. Une copie complète d'un instantané conserve toutes les données et métadonnées requises pour restaurer l'instance de base de données. Après la première copie instantanée, vous pouvez copier des instantanés incrémentiels de la même instance de base de données vers la même région de destination au sein du même AWS compte.

Un instantané incrémentiel contient uniquement les données qui ont été modifiées après l'instantané le plus récent de la même instance de base de données. La copie d'instantanés incrémentiels est plus rapide est entraîne des coûts de stockage plus faibles que la copie d'instantanés complets. La copie incrémentielle d'instantanés entre AWS régions est prise en charge pour les instantanés chiffrés et non chiffrés.

> **Important**
>
> Pour les instantanés partagés, la copie d'instantanés incrémentiels n'est pas prise en charge. Pour les instantanés partagés, toutes les copies sont des instantanés complets, même au sein de la même région.

En fonction des AWS régions concernées et de la quantité de données à copier, la réalisation d'une copie instantanée entre régions peut prendre des heures.

Copier un instantané de cluster de base de données à l'aide de la console

Si votre moteur de base de données source est Neptune, l'instantané sera un instantané de cluster de bases de données. Pour chaque AWS compte, vous pouvez copier jusqu'à cinq instantanés de cluster de base de données à la fois par AWS région. La copie des instantanés de cluster de base données chiffrés et non chiffrés est prise en charge.

Pour plus d'informations sur la tarification du transfert des données, consultez la section Tarification de Neptune.

Pour annuler une opération de copie une fois qu'elle est en cours, supprimez l'instantané de cluster de bases de données cible pendant que cet instantané de cluster de bases de données a le statut copying (copie en cours).

La procédure suivante permet de copier des instantanés de cluster de base de données chiffrés ou non chiffrés.

Pour copier un instantané de cluster de bases de données

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à l'adresse. https://console.aws.amazon.com/neptune/home
2. Dans le panneau de navigation, choisissez Snapshots.
3. Cochez la case correspondant à l'instantané de cluster de bases de données automatisé que vous souhaitez copier.
4. Choisissez Actions, puis Copy Snapshot (Copier l'instantané). La page Make Copy of DB Snapshot (Faire une copie d'instantanés de base de données) apparaît.
5. Saisissez le nom de la copie de l'instantané de cluster de bases de données dans Nouvel identifiant d'instantané de base de données.
6. Pour copier les balises et les valeurs de l'instantané vers la copie de cet instantané, choisissez Copy Tags (Copier les balises).
7. Pour Enable Encryption (Activer le chiffrement), choisissez l'une des options suivantes :
   - Si l'instantané de cluster de bases de données n'est pas chiffré et que vous ne souhaitez pas chiffrer la copie, choisissez Disable encryption (Désactiver le chiffrement).
   - Si l'instantané de cluster de bases de données n'est pas chiffré mais que vous souhaitez chiffrer la copie, choisissez Enable encryption (Activer le chiffrement). Dans ce cas, pour Master Key, spécifiez l'identifiant de AWS KMS clé à utiliser pour chiffrer la copie instantanée du cluster de base de données.
   - Si l'instantané de cluster de bases de données est chiffré, choisissez Enable encryption (Activer le chiffrement). Dans ce cas, vous devez chiffrer la copie. Oui est donc déjà sélectionné. Pour Master Key, spécifiez l'identifiant de AWS KMS clé à utiliser pour chiffrer la copie instantanée du cluster de base de données.
8. Choisissez Copier l'instantané.

Copier un instantané de cluster de base de données à l'aide du AWS CLI

Vous pouvez copier un instantané de base de données en utilisant la commande copy-db-cluster-snapshot de l' AWS CLI .

Si vous copiez l'instantané dans une nouvelle AWS région, exécutez la commande dans la nouvelle région.

Utilisez les descriptions et exemples de paramètres suivants pour déterminer les paramètres à utiliser pour copier un instantané avec l' AWS CLI.

- `--source-db-cluster-snapshot-identifier` : identifiant de l'instantané de base de données source.
  - Si l'instantané source se trouve dans la même AWS région que la copie, spécifiez un identifiant d'instantané de base de données valide, par exemple`neptune:instance1-snapshot-20130805`.
  - Si l'instantané source se trouve dans une AWS région différente de celle de la copie, spécifiez un ARN d'instantané de base de données valide tel que`arn:aws:neptune:us-west-2:123456789012:snapshot:instance1-snapshot-20130805`.
  - Si vous effectuez la copie à partir d'un instantané de base de données manuel partagé, ce paramètre doit être l'Amazon Resource Name (ARN) de l'instantané de base de données partagé.
  - Si vous copiez un instantané chiffré, ce paramètre doit être au format ARN de la AWS région source et doit correspondre `SourceDBSnapshotIdentifier` à celui du `PreSignedUrl` paramètre.
- `--target-db-cluster-snapshot-identifier` : identifiant de la nouvelle copie de l'instantané de base de données chiffré.
- `--kms-key-id`— — L'ID de AWS KMS clé pour un instantané de base de données chiffré. L'ID de AWS KMS clé est le Amazon Resource Name (ARN), AWS KMS l'identifiant de AWS KMS clé ou l'alias de clé de AWS KMS chiffrement.
  - Si vous copiez un instantané de base de données chiffré depuis votre AWS compte, vous pouvez spécifier une valeur pour ce paramètre afin de chiffrer la copie avec une nouvelle clé de AWS KMS chiffrement. Si vous ne spécifiez aucune valeur pour ce paramètre, la copie de l'instantané de base de données est cryptée avec la même AWS KMS clé que l'instantané de base de données source.

  - Vous ne pouvez pas utiliser ce paramètre pour créer une copie chiffrée d'un instantané non chiffré. Si vous essayez de le faire, une erreur se produira.
  - Si vous copiez un instantané chiffré dans une autre AWS région, vous devez spécifier une AWS KMS clé pour la AWS région de destination. AWS KMS les clés de chiffrement sont spécifiques à la AWS région dans laquelle elles sont créées, et vous ne pouvez pas utiliser les clés de chiffrement d'une AWS région dans une autre AWS région.
- `--source-region`— — L'ID de la AWS région où se trouve l'instantané de base de données source. Si vous copiez un instantané chiffré dans une autre AWS région, vous devez spécifier cette option.
- `--region`— — L'ID de la AWS région dans laquelle vous copiez l'instantané. Si vous copiez un instantané chiffré dans une autre AWS région, vous devez spécifier cette option.

Example Source non chiffrée, même région de destination

Le code suivant crée une copie d'un instantané, portant le nouveau nom`mydbsnapshotcopy`, d'une `us-east-1` AWS région à une `us-west-2` autre.

Pour Linux, OS X ou Unix :

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

Pour Windows :

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

Example Source non chiffrée, autre région de destination

Le code suivant crée une copie d'un instantané, portant le nouveau nom`mydbsnapshotcopy`, d'une `us-east-1` AWS région à une `us-west-2` autre. Exécutez la commande dans la région `us-west-2`.

Pour Linux, OS X ou Unix :

```
aws neptune copy-db-cluster-snapshot \
```

```
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
east-1:123456789012:snapshot:instance1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy \
  --source-region us-east-1 \
  --region us-west-2
```

Pour Windows :

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
east-1:123456789012:snapshot:instance1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy ^
  --source-region us-east-1 ^
  --region us-west-2
```

Example Source chiffrée, autre région de destination

L'exemple de code suivant copie un instantané de base de données chiffré de la `us-east-1` AWS région vers la `us-west-2` région. Exécutez la commande dans la région `us-west-2`.

Pour Linux, OS X ou Unix :

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
west-2:123456789012:snapshot:instance1-snapshot-20161115 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy \
  --source-region us-east-1 \
  --region us-west-2
  --kms-key-id my_us_west_2_key
```

Pour Windows :

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
west-2:123456789012:snapshot:instance1-snapshot-20161115 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy ^
  --source-region us-east-1 ^
  --region us-west-2
  --kms-key-id my-us-west-2-key
```

Partage d'un instantané de cluster de base de données

Neptune vous permet de partager un instantané de cluster de bases de données manuel de différentes façons :

- Le partage manuel d'un instantané de cluster de base de données, qu'il soit chiffré ou non, permet aux AWS comptes autorisés de copier l'instantané.
- Le partage d'un instantané de cluster de base de données manuel non chiffré permet aux AWS comptes autorisés de restaurer directement un cluster de base de données à partir de l'instantané au lieu d'en prendre une copie et de le restaurer à partir de celui-ci. Les instantanés chiffrés ne peuvent pas être restaurés directement ; ils doivent d'abord être copiés, puis restaurés à partir de la copie.

> **Note**
>
> Pour partager un instantané de cluster de bases de données automatisé, créez un instantané de cluster de bases de données manuel en copiant l'instantané automatisé, puis partagez cette copie.

Pour plus d'informations sur la restauration d'un cluster de base de données à partir d'un instantané de cluster de base de données, consultez Comment effectuer une restauration à partir d'un instantané.

Vous pouvez partager un instantané manuel avec un maximum de 20 autres AWS comptes. Vous pouvez également partager un instantané manuel non chiffré en tant que public, ce qui le rend accessible à tous les AWS comptes. Lors du partage d'un instantané marqué comme public, n'incluez aucune information privée dans vos instantanés publics.

> **Note**
>
> Lorsque vous restaurez un cluster de base de données à partir d'un instantané partagé à l'aide de l'API AWS Command Line Interface (AWS CLI) ou Neptune, vous devez spécifier le nom de ressource Amazon (ARN) du cliché partagé comme identifiant de l'instantané.

Rubriques

- Partage d'un instantané de cluster de base de données chiffré
- Partage d'un instantané de cluster de base de données

Partage d'un instantané de cluster de base de données chiffré

Vous pouvez partager des instantanés de cluster de base de données qui ont été chiffrés « au repos » en utilisant l'algorithme de chiffrement AES-256. Pour de plus amples informations, veuillez consulter Chiffrer les ressources Neptune au repos. Pour ce faire, vous devez exécuter les étapes suivantes :

1. Partagez la clé de chiffrement AWS Key Management Service (AWS KMS) qui a été utilisée pour chiffrer l'instantané avec tous les comptes auxquels vous souhaitez accéder à l'instantané.

   Vous pouvez partager des clés de AWS KMS chiffrement avec un autre AWS compte en ajoutant cet autre compte à la politique des clés KMS. Pour plus de détails sur la mise à jour d'une stratégie de clé, voir Stratégies de clés dans le Guide du développeur AWS KMS . Pour obtenir un exemple de création d'une stratégie de clé, consultez Création d'une stratégie IAM pour permettre la copie d'un instantané chiffré plus loin dans cette rubrique.
2. Utilisez l'API AWS Management Console AWS CLI, ou Neptune pour partager l'instantané chiffré avec les autres comptes.

Ces restrictions s'appliquent au partage d'instantanés chiffrés :

- Vous ne pouvez pas partager des instantanés chiffrés comme publics.
- Vous ne pouvez pas partager un instantané chiffré à l'aide de la clé de AWS KMS chiffrement par défaut du AWS compte qui a partagé l'instantané.

Autoriser l'accès à une clé AWS KMS de chiffrement

Pour qu'un autre AWS compte puisse copier un instantané de cluster de base de données chiffré partagé depuis votre compte, le compte avec lequel vous partagez votre instantané doit avoir accès à la clé KMS qui a chiffré l'instantané. Pour autoriser un autre AWS compte à accéder à une AWS KMS clé, mettez à jour la politique de clé pour la clé KMS avec l'ARN du AWS compte avec lequel vous partagez, comme indiqué `Principal` dans la politique de clé KMS. Autorisez ensuite l'action `kms:CreateGrant`. Pour obtenir des instructions générales, consultez Autoriser des utilisateurs

d'autres comptes à utiliser une clé KMS dans le Guide du développeur AWS Key Management Service .

Une fois que vous avez accordé à un AWS compte l'accès à votre clé de chiffrement KMS, pour copier votre instantané chiffré, ce AWS compte doit créer un utilisateur IAM s'il n'en possède pas déjà un. Les restrictions de sécurité KMS n'autorisent pas l'utilisation d'une identité de AWS compte root pour cela. Le AWS compte doit également associer une politique IAM à cet utilisateur IAM qui lui permet de copier un instantané de cluster de base de données chiffré à l'aide de votre clé KMS.

Dans l'exemple de stratégie suivant, l'utilisateur `111122223333` est le propriétaire de la clé de chiffrement KMS, et l'utilisateur `444455556666` est le compte avec lequel la clé est partagée. Cette politique clé mise à jour permet au AWS compte d'accéder à la clé KMS en incluant l'ARN pour l'identité du AWS compte racine de l'utilisateur en `444455556666` tant `Principal` que politique, et en autorisant l'`kms:CreateGrant`action.

JSON

```
{
    "Id": "key-policy-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::444455556666:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "AllowAttachmentOfPersistentResources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::444455556666:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

Création d'une stratégie IAM pour permettre la copie d'un instantané chiffré

Une fois que le AWS compte externe a accès à votre clé KMS, le propriétaire de ce compte peut créer une politique permettant à un utilisateur IAM créé pour le compte de copier un instantané chiffré avec cette clé KMS.

L'exemple suivant illustre une stratégie qui est peut être attachée à un utilisateur IAM pour le compte AWS `444455556666`. Ceci permet à l'utilisateur IAM de copier un instantané partagé depuis le compte AWS `111122223333` qui a été chiffré avec la clé KMS `c989c1dd-a3f2-4a5d-8d96-e793d082ab26` dans la région `us-west-2`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUseOfTheKey",
```

```
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey",
                "kms:CreateGrant",
                "kms:RetireGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"]
        },
        {
            "Sid": "AllowAttachmentOfPersistentResources",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"],
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

Pour plus de détails sur la mise à jour d'une stratégie de clé, voir Stratégies de clés dans le Guide du développeur AWS Key Management Service .

Partage d'un instantané de cluster de base de données

Vous pouvez partager un instantané de cluster de base de données à l' AWS Management Console aide de l'API AWS CLI Neptune ou de l'API Neptune.

Utilisation de la console pour partager un instantané de cluster de base de données

En utilisant la console Neptune, vous pouvez partager un instantané de cluster de bases de données manuel avec jusqu'à 20 comptes AWS . Vous pouvez également arrêter le partage d'un instantané manuel avec un ou plusieurs comptes.

Pour partager un instantané de cluster de base de données manuel

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le panneau de navigation, choisissez Snapshots (Instantanés).
3. Sélectionnez l'instantané manuel que vous souhaitez partager.
4. Choisissez Actions, Share Snapshot (Partager l'instantané).
5. Choisissez l'une des options suivantes pour DB snapshot visibility (Visibilité d'instantané de base de données).

   - Si la source est non chiffrée, choisissez Public pour permettre à tous les comptes AWS de restaurer un cluster de bases de données à partir de l'instantané de cluster de bases de données manuel. Vous pouvez également choisir Privé pour autoriser uniquement les AWS comptes que vous spécifiez à restaurer un cluster de base de données à partir de votre instantané de cluster de base de données manuel.

     > **Warning**
     >
     > Si vous définissez la visibilité des instantanés de base de données sur Public, tous les AWS comptes peuvent restaurer un cluster de base de données à partir de votre instantané de cluster de base de données manuel et avoir accès à vos données. Ne partagez pas en Public un instantané manuel de cluster DB contenant des informations privées.

   - Si la source est chiffrée, la Visibilité d'instantané de base de données est définie sur Privé, car les instantanés chiffrés ne peuvent pas être partagés s'ils sont marqués comme étant publics.
6. Pour ID de AWS compte, entrez l'identifiant du AWS compte que vous souhaitez autoriser à restaurer un cluster de bases de données à partir de votre instantané manuel. Choisissez ensuite Ajouter. Répétez l'opération pour inclure des identifiants de AWS compte supplémentaires, jusqu'à 20 AWS comptes.

Si vous commettez une erreur lors de l'ajout d'un identifiant de AWS compte à la liste des comptes autorisés, vous pouvez le supprimer de la liste en choisissant Supprimer à droite de l'identifiant de AWS compte incorrect.

7. Après avoir ajouté des identifiants pour tous les AWS comptes que vous souhaitez autoriser à restaurer l'instantané manuel, choisissez Enregistrer.

Pour arrêter de partager un instantané manuel de cluster de base de données avec un AWS compte

1. Ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le panneau de navigation, choisissez Snapshots (Instantanés).
3. Sélectionnez l'instantané manuel que vous souhaitez cesser de partager.
4. Choisissez Actions, puis Share Snapshot (Partager instantané).
5. Pour supprimer l'autorisation d'un AWS compte, choisissez Supprimer comme identifiant de AWS compte pour ce compte dans la liste des comptes autorisés.
6. Choisissez Enregistrer.

Suppression d'un instantané Neptune

Vous pouvez supprimer un instantané de base de données à l' AWS Management Console aide de l'API de gestion AWS CLI, de, ou de Neptune :

Suppression à l'aide de la console

1. Connectez-vous à la console AWS de gestion et ouvrez la console Amazon Neptune à https://console.aws.amazon.com/neptune/ la maison.
2. Dans le panneau de navigation, choisissez Snapshots (Instantanés).
3. Choisissez l'instantané de base de données à supprimer.
4. Pour Actions, choisissez Supprimer la pile.
5. Dans la page de confirmation, sélectionnez Supprimer.

Suppression à l'aide du AWS CLI

Vous pouvez également supprimer un instantané de base de données à l'aide de la commande de l' AWS CLI delete_db_cluster_snapshot en utilisant le paramètre `--db-snapshot-identifier` pour identifier l'instantané à supprimer :

Pour Linux, OS X ou Unix :

```
aws neptune delete-db-cluster-snapshot \
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

Pour Windows :

```
aws neptune delete-db-cluster-snapshot ^
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

Suppression à l'aide de l'API de gestion Neptune

Vous pouvez utiliser l'un des SDKs pour supprimer un instantané de base de données en appelant l'Supprimer DBCluster un instantanéAPI et en utilisant les `DBSnapshotIdentifier` paramètres pour identifier l'instantané de base de données à supprimer.

Bonnes pratiques : optimisation de Neptune

Voici quelques recommandations d'ordre général pour utiliser Amazon Neptune. Utilisez ces informations comme référence pour trouver rapidement des recommandations sur l'utilisation d'Amazon Neptune et l'optimisation des performances.

Table des matières

- Directives opérationnelles de base Amazon Neptune
  - Bonnes pratiques de sécurité pour Amazon Neptune
  - Éviter différentes classes d'instances dans un cluster
  - Éviter les redémarrages répétés pendant le chargement en bloc
  - Activation de l'index OSGP si le nombre de prédicats est élevé
  - Éviter les transactions de longue durée dans la mesure du possible
  - Bonnes pratiques pour l'utilisation des métriques Neptune
  - Bonnes pratiques pour le réglage des requêtes Neptune
  - Équilibrage de charge entre les réplicas en lecture
  - Chargement plus rapide à l'aide d'une instance temporaire de plus grande taille
  - Redimensionnement de l'instance d'enregistreur en basculant vers un réplica en lecture
  - Nouvelle tentative de chargement après une erreur d'interruption de tâche de lecture anticipée des données
- Bonnes pratiques d'ordre général pour l'utilisation de Gremlin avec Neptune
  - Configuration Heartbeat pour Neptune Serverless
  - Structurer les requêtes upsert pour tirer parti du moteur DFE
  - Testez le code Gremlin dans le contexte dans lequel vous allez le déployer
  - Création d'écritures Gremlin multithreads efficaces
  - Élagage des enregistrements avec la propriété de date/heure de création
  - Utilisation de la méthode datetime () pour Groovy Time Data
  - Utilisation de la date et de l'heure natives pour les données temporelles GLV
- Bonnes pratiques pour l'utilisation du client Java Gremlin avec Neptune
  - Réutilisation de l'objet client dans plusieurs threads
  - Création d'objets client Java Gremlin distincts pour les points de terminaison en lecture et en écriture

- Ajout de plusieurs points de terminaison de réplica en lecture à un groupe de connexions Java Gremlin
- Fermeture du client pour éviter de limiter les connexions
- Création d'une connexion après un basculement
- Définition de maxInProcessPerConnection et maxSimultaneousUsagePerConnection sur la même valeur
- Envoi de requêtes au serveur sous la forme de bytecode et non de chaînes
- Consommez toujours complètement l'itérateur ResultSet ou l'itérateur renvoyé par une requête
- Ajout en bloc de sommets et d'arêtes dans des lots
- Désactivation de la mise en cache du DNS dans la machine virtuelle Java
- Définition facultative de délais d'expiration au niveau de chaque requête
- Résolution des problèmes de java.util.concurrent.TimeoutException

- Bonnes pratiques Neptune avec openCypher et Bolt
  - Création d'une connexion après un basculement
  - Gestion des connexions pour les applications de longue durée
  - Gestion des connexions pour AWS Lambda
  - Préférer les arêtes dirigées aux arêtes bidirectionnelles dans les requêtes
  - Neptune ne prend pas en charge plusieurs requêtes simultanées dans une transaction
  - Fermer les objets Driver lorsque vous avez terminé
  - Utilisation des modes de transaction explicites pour la lecture et l'écriture
    - Transactions en lecture seule
    - Transactions de mutation
  - Logique des nouvelles tentatives pour les exceptions
  - Définissez plusieurs propriétés à la fois à l'aide d'une seule clause SET
    - Utilisez la clause SET pour supprimer plusieurs propriétés à la fois
  - Utilisation des requêtes paramétrées
  - Utilisez des cartes aplaties au lieu de cartes imbriquées dans la clause UNWIND
  - Placez des nœuds plus restrictifs sur le côté gauche dans les expressions VLP (Variable-Length Path)
  - Évitez les vérifications redondantes des étiquettes des nœuds en utilisant des noms de relations granulaires

  - Spécifiez les étiquettes de bord dans la mesure du possible
  - Évitez d'utiliser la clause WITH dans la mesure du possible
  - Placez les filtres restrictifs le plus tôt possible dans la requête
  - Vérifiez explicitement si les propriétés existent
  - N'utilisez pas de chemin nommé (sauf si cela est obligatoire)
  - Évitez COLLECT (DISTINCT ())
  - Préférez la fonction de propriétés à la recherche de propriétés individuelle lors de la récupération de toutes les valeurs de propriété
  - Effectuer des calculs statiques en dehors de la requête
  - Entrées par lots utilisant UNWIND au lieu d'instructions individuelles
  - Préférez utiliser la personnalisation IDs pour le nœud ou la relation
  - Évitez de faire des calculs ~id dans la requête
  - Mise à jour/fusion de plusieurs nœuds
- Bonnes pratiques Neptune avec l'utilisation de SPARQL
  - Interrogation de tous les graphes par défaut
  - Spécification d'un graphe nommé pour le chargement
  - Choisir entre FILTER, FILTER...IN et VALUES dans vos requêtes
- Bonnes pratiques d'utilisation de Neptune avec Amazon ECS et Amazon EKS
  - Gérez les pools de connexions et le cycle de vie dans les environnements conteneurisés
  - Gérer le remplacement de l'hôte et le blocage de la connexion
  - Configuration du réseau, des groupes de sécurité et de l'authentification IAM
- Bonnes pratiques d'utilisation du AWS CLI et SDK avec Neptune

Directives opérationnelles de base Amazon Neptune

Vous trouverez ci-dessous les directives opérationnelles de base que vous devez suivre lorsque vous utilisez Neptune.

- Familiarisez-vous avec les instances de base de données Neptune afin de pouvoir les dimensionner de manière appropriée en fonction de vos exigences en matière de performances et de cas d'utilisation. Consultez Clusters de bases de données et instances de base de données Amazon Neptune.

- Surveillez votre utilisation de CPU et de la mémoire. Ceci vous permet de savoir quand migrer vers une classe d'instances de base de données avec une plus grande capacité de CPU ou de la mémoire pour atteindre les performances des requêtes dont vous avez besoin. Vous pouvez configurer Amazon CloudWatch pour qu'il vous avertisse lorsque les habitudes d'utilisation changent ou lorsque vous approchez de la capacité de votre déploiement. Ceci peut vous aider à maintenir les performances et la disponibilité du système. Consultez Surveillance des instances et Surveillance de Neptune pour en savoir plus.

  Comme Neptune dispose de son propre gestionnaire de mémoire, il est normal d'observer une utilisation de mémoire relativement faible, et ce même en cas de forte utilisation de CPU. Le fait de rencontrer out-of-memory des exceptions lors de l'exécution de requêtes est le meilleur indicateur de la nécessité d'augmenter la mémoire disponible.
- Activez les sauvegardes automatiques et définissez la fenêtre de sauvegarde pour qu'elles se produisent à un moment opportun.
- Testez le basculement pour votre instance de base de données afin de connaître la durée du processus pour votre cas d'utilisation. Il permet également de veiller à ce que l'application qui accède à votre instance DB puisse automatiquement se connecter à la nouvelle instance DB suite au basculement.
- Si possible, exécutez votre client et votre cluster Neptune dans la même région et le même VPC, car les connexions entre régions avec l'appairage de VPC peuvent entraîner des délais dans les temps de réponse des requêtes. Pour obtenir des réponses aux requêtes en moins de 10 millisecondes, il est nécessaire de conserver le client et le cluster Neptune dans la même région et le même VPC.
- Lorsque vous créez une instance de réplica en lecture, elle doit être au moins aussi volumineuse que l'instance d'enregistreur principale. Cela permet de contrôler le retard de réplication et d'éviter les redémarrages du réplica. Consultez Éviter différentes classes d'instances dans un cluster.
- Avant de procéder à une mise à niveau vers une nouvelle version majeure du moteur, assurez-vous de tester votre application sur cette dernière. Pour ce faire, clonez le cluster de bases de données afin qu'il exécute la nouvelle version du moteur, puis testez votre application sur ce clone.
- Pour faciliter les basculements, toutes les instances devraient idéalement avoir la même taille.

Rubriques

- Bonnes pratiques de sécurité pour Amazon Neptune
- Éviter différentes classes d'instances dans un cluster
- Éviter les redémarrages répétés pendant le chargement en bloc

- Activation de l'index OSGP si le nombre de prédicats est élevé
- Éviter les transactions de longue durée dans la mesure du possible
- Bonnes pratiques pour l'utilisation des métriques Neptune
- Bonnes pratiques pour le réglage des requêtes Neptune
- Équilibrage de charge entre les réplicas en lecture
- Chargement plus rapide à l'aide d'une instance temporaire de plus grande taille
- Redimensionnement de l'instance d'enregistreur en basculant vers un réplica en lecture
- Nouvelle tentative de chargement après une erreur d'interruption de tâche de lecture anticipée des données

Bonnes pratiques de sécurité pour Amazon Neptune

Utilisez des comptes Gestion des identités et des accès AWS (IAM) pour contrôler l'accès aux actions de l'API Neptune. Contrôlez les actions qui créent, modifient ou suppriment les ressources Neptune (comme les instances de base de données, les groupes de sécurité, les groupes d'options ou les groupes de paramètres), et celles qui effectuent des tâches administratives courantes (comme la sauvegarde et la restauration d'instances de base de données).

- Utilisez des informations d'identification temporaires plutôt que persistantes dans la mesure du possible.
- Attribuez un compte IAM individuel à chaque personne qui gère les ressources Amazon Relational Database Service (Amazon RDS). N'utilisez jamais AWS d'utilisateurs root pour gérer les ressources Neptune. Créez un utilisateur IAM pour chaque personne, y compris vous-même.
- Accordez à chaque utilisateur un ensemble minimum d'autorisations requises pour exécuter ses tâches.
- Utilisez des groupes IAM pour gérer efficacement des autorisations pour plusieurs utilisateurs.
- Effectuer une rotation régulière des informations d'identification IAM.

Pour plus d'informations sur l'utilisation d'IAM afin d'accéder aux ressources Neptune, consultez Sécurisation de votre base de données Amazon Neptune. Pour obtenir des informations générales sur l'utilisation d'IAM, consultez Gestion des identités et des accès AWS et Bonnes pratiques IAM dans le Guide de l'utilisateur IAM.

Éviter différentes classes d'instances dans un cluster

Lorsque votre cluster de bases de données contient des instances de différentes classes, des problèmes peuvent survenir au fil du temps. Le problème le plus courant est qu'une petite instance de lecteur peut entrer dans un cycle de redémarrages répétés en raison d'un retard de réplication. Si la configuration de la classe d'instances de base de données d'un nœud de lecture est plus faible que celle d'une instance de base de données d'enregistreur, le volume de modifications peut être trop vaste pour que le lecteur puisse tout gérer.

> **Important**
>
> Pour éviter les redémarrages répétés provoqués par un retard de réplication, configurez le cluster de bases de données de manière à ce que toutes les instances aient la même classe (taille) d'instance.

Vous pouvez voir le décalage entre l'instance du rédacteur (l'instance principale) et les lecteurs de votre cluster de bases de données à l'aide de la `ClusterReplicaLag` métrique d'Amazon CloudWatch. La métrique `VolumeWriteIOPs` vous permet également de détecter les pics d'activité d'écriture du cluster susceptibles de provoquer un retard de réplication.

Éviter les redémarrages répétés pendant le chargement en bloc

Si vous êtes confronté à un cycle de redémarrages répétés des réplicas en lecture en raison d'un retard de réplication lors d'un chargement en bloc, les réplicas ne parviendront probablement pas à suivre le rythme de l'enregistreur du cluster de bases de données.

Vous pouvez soit mettre à l'échelle les lecteurs pour qu'ils soient plus grands que l'enregistreur, soit les supprimer temporairement pendant le chargement en bloc, puis les recréer une fois le chargement terminé.

Activation de l'index OSGP si le nombre de prédicats est élevé

Si le modèle de données contient un grand nombre de prédicats distincts (plus d'un millier dans la plupart des cas), vous pouvez subir une baisse des performances et une augmentation des coûts d'exploitation.

Si tel est le cas, vous pouvez activer l'index OSGP pour améliorer les performances. Consultez Index OSGP.

Éviter les transactions de longue durée dans la mesure du possible

Les transactions de longue durée, qu'elles soient en lecture seule ou en lecture-écriture, peuvent causer des problèmes inattendus tels que les suivants :

Une transaction de longue durée sur une instance de lecteur ou d'enregistreur avec des écritures simultanées peut entraîner une accumulation importante de versions de données différentes. Les temps de latence peuvent être plus élevés pour les requêtes de lecture qui filtrent une grande partie de leurs résultats.

Dans certains cas, les versions accumulées au fil des heures peuvent entraîner la limitation des nouvelles écritures.

Une transaction de lecture-écriture de longue durée impliquant de nombreuses écritures peut également causer des problèmes si l'instance redémarre. Si une instance redémarre à la suite d'un événement de maintenance ou d'un blocage, toutes les écritures non validées seront annulées. Ces opérations d'annulation s'exécutent généralement en arrière-plan et n'empêchent pas l'instance de se rétablir, mais toute nouvelle écriture entrant en conflit avec les opérations en cours d'annulation échouera.

Par exemple, si la même requête fait l'objet d'une nouvelle tentative après une interruption de la connexion lors de l'exécution précédente, elle pourra échouer au redémarrage de l'instance.

Le temps nécessaire aux opérations d'annulation est proportionnel à l'ampleur des modifications impliquées.

Bonnes pratiques pour l'utilisation des métriques Neptune

Pour identifier les problèmes de performances causés par des ressources insuffisantes et d'autres goulots d'étranglement courants, vous pouvez surveiller les métriques disponibles pour votre cluster de bases de données Neptune.

Surveillez régulièrement les métriques de performances pour rassembler les données sur les valeurs moyennes, maximales et minimales pour différents intervalles de temps. Cela vous aide à déterminer quand les performances se dégradent. À l'aide de ces données, vous pouvez définir des CloudWatch alarmes Amazon pour des seuils métriques spécifiques afin d'être alerté s'ils sont atteints.

Lorsque vous configurez un nouveau cluster de base de données et l'exécutez avec une charge de travail classique, essayez de capturer les valeurs moyennes, maximales et minimales de toutes les

métriques de performances à plusieurs intervalles différents (par exemple, une heure, 24 heures, une semaine, deux semaines). Cela vous permet de vous faire une idée de ce qui est normal. Cela permet de comparer l'activité pendant les heures pleines et les heures creuses. Vous pouvez alors utiliser ces informations pour identifier les périodes où les performances ont chuté en dessous des niveaux standard et définir des alarmes en conséquence.

Consultez Surveillance de Neptune à l'aide d'Amazon CloudWatch pour découvrir comment afficher les métriques Neptune.

Voici les métriques les plus importantes pour démarrer :

- BufferCacheHitRatio— Le pourcentage de demandes traitées par le cache tampon. Les échecs d'accès au cache ajoutent une latence importante à l'exécution des requêtes. Si le taux d'accès au cache est inférieur à 99,9 % et que la latence constitue un problème pour votre application, envisagez de mettre à niveau le type d'instance afin de mettre en cache davantage de données en mémoire.
- Utilisation de CPU : pourcentage de la capacité de traitement informatique utilisée. Selon vos objectifs en matière de performance des requêtes, des valeurs de consommation de CPU élevées peuvent être appropriées.
- Mémoire libérable : quantité de RAM disponible sur l'instance de base de données, en octets. Neptune possède son propre gestionnaire de mémoire. Cette métrique peut donc être inférieure à ce à quoi vous vous attendiez. Si les requêtes génèrent souvent des out-of-memory exceptions, vous devriez envisager de mettre à niveau votre classe d'instance vers une classe dotée de plus de RAM.

La ligne rouge sous l'onglet Surveillance indique 75 % pour les métriques de CPU et de mémoire. Si la consommation de mémoire de l'instance franchit régulièrement cette ligne, vérifiez votre charge de travail et envisagez de mettre à niveau votre instance pour améliorer les performances des requêtes.

Bonnes pratiques pour le réglage des requêtes Neptune

L'un des meilleurs moyens d'améliorer les performances de Neptune consiste à ajuster les requêtes les plus communément utilisées et exigeantes en ressources pour les rendre moins onéreuses à exécuter.

Pour en savoir plus sur le réglage des requêtes Gremlin, consultez Indicateurs de requête Gremlin et Réglage des requêtes Gremlin. Pour plus d'informations sur la façon d'ajuster les requêtes SPARQL, consultez Indicateurs de requête SPARQL.

Équilibrage de charge entre les réplicas en lecture

Le routage en tourniquet du point de terminaison de lecteur modifie l'hôte vers lequel l'entrée DNS pointe. Le client doit créer une nouvelle connexion et résoudre l'enregistrement DNS pour obtenir une connexion à une nouvelle réplique en lecture, car WebSocket les connexions sont souvent maintenues actives pendant de longues périodes.

Pour obtenir différents réplicas en lecture pour les requêtes successives, assurez-vous que votre client résout l'entrée DNS chaque fois qu'il se connecte. Cela peut demander la fermeture de la connexion et la reconnexion au point de terminaison du lecteur.

Vous pouvez également équilibrer la charge des requêtes entre des réplicas en lecture en vous connectant à des points de terminaison d'instance de manière explicite.

Chargement plus rapide à l'aide d'une instance temporaire de plus grande taille

Vos performances de chargement augmentent avec des tailles d'instance plus grandes. Si vous n'utilisez pas un grand type d'instance, mais que vous souhaitez accélérer le chargement, vous pouvez utiliser une instance plus grande pour le chargement, puis la supprimer.

> **Note**
>
> La procédure suivante est pour un nouveau cluster. Si vous disposez déjà d'un cluster existant, vous pouvez ajouter une nouvelle instance plus grande, puis la promouvoir en instance de base de données principale.

Pour charger des données à l'aide d'une taille d'instance plus grande

1. Créez un cluster avec une seule instance `r5.12xlarge`. Cette instance est l'instance de base de données principale.
2. Créez un ou plusieurs réplicas en lecture de même taille (`r5.12xlarge`).

   Vous pouvez créer des réplicas en lecture de plus petite taille, mais s'ils ne sont pas assez grands pour suivre le rythme des écritures effectuées par l'instance principale, ils nécessiteront peut-être des redémarrages fréquents. Les temps d'arrêt qui en résulteront réduiront considérablement les performances.

3. Dans la commande de chargement en bloc, incluez `“parallelism” : “OVERSUBSCRIBE”` pour indiquer à Neptune d'utiliser toutes les ressources de CPU disponibles pour le chargement (voir Paramètres de demande du chargeur Neptune). L'opération de chargement se déroulera alors aussi rapidement que possible, I/O ce qui nécessite généralement 60 à 70 % des ressources du processeur.
4. Chargez vos données à l'aide du chargeur Neptune. La tâche de chargement s'exécute sur l'instance de base de données principale.
5. Une fois le chargement des données terminé, veillez à réduire toutes les instances du cluster au même type d'instance afin d'éviter des frais supplémentaires et des problèmes de redémarrage répétés (voir Éviter différentes tailles d'instance).

Redimensionnement de l'instance d'enregistreur en basculant vers un réplica en lecture

La meilleure façon de redimensionner une instance de votre cluster de bases de données, y compris l'instance d'enregistreur, consiste à créer ou à modifier une instance de réplica en lecture afin qu'elle ait la taille souhaitée, puis de basculer délibérément vers ce réplica. Le temps d'arrêt observé par votre application correspond uniquement au temps nécessaire pour modifier l'adresse IP de l'enregistreur. Il devrait être d'environ trois à cinq secondes.

L'API de gestion Neptune que vous utilisez pour basculer délibérément de l'instance d'enregistreur actuelle vers une instance de réplica en lecture est Basculement DBCluster. Si vous utilisez le client Java Gremlin, vous devrez peut-être créer un objet client après le basculement pour relever la nouvelle adresse IP, comme indiqué ici.

Assurez-vous d'ajuster vos instances pour qu'elles aient toutes la même taille afin d'éviter un cycle de redémarrages répétés, comme indiqué ci-dessous.

Nouvelle tentative de chargement après une erreur d'interruption de tâche de lecture anticipée des données

Lorsque vous chargez des données dans Neptune à l'aide du chargeur en bloc, vous pouvez parfois obtenir le statut `LOAD_FAILED` avec une erreur `PARSING_ERROR`, tandis que le message `Data prefetch task interrupted` peut être indiqué en réponse à une demande d'informations détaillées, comme suit :

```
"errorLogs" : [
```

```
  {
    "errorCode" : "PARSING_ERROR",
    "errorMessage" : "Data prefetch task interrupted: Data prefetch task for 11467
 failed",
    "fileName" : "s3://amzn-s3-demo-bucket/some-source-file",
    "recordNum" : 0
  }
]
```

Si vous rencontrez cette erreur, essayez simplement de relancer la demande de téléchargement par lot.

L'erreur se produit en raison d'une interruption temporaire qui n'est généralement pas provoquée par votre demande ni vos données, et elle peut généralement être résolue en réexécutant la demande de chargement par lot.

Si vous utilisez les paramètres par défaut, à savoir `"mode":"AUTO"` et `"failOnError":"TRUE"`, le chargeur ignore les fichiers qu'il a déjà chargés avec succès et reprend le chargement des fichiers qu'il n'avait pas encore chargés lorsque l'interruption s'est produite.

Bonnes pratiques d'ordre général pour l'utilisation de Gremlin avec Neptune

Suivez ces recommandations lorsque vous utilisez le langage de parcours de graphe Gremlin avec Neptune. Pour plus d'informations sur l'utilisation de Gremlin avec Neptune, consultez the section called “Gremlin”.

> **Important**
>
> Une modification a été apportée à TinkerPop la version 3.4.11 qui améliore l'exactitude du traitement des requêtes, mais qui, pour le moment, peut parfois avoir un impact sérieux sur les performances des requêtes.
> Par exemple, une requête de ce type peut être beaucoup plus lente :
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
>   out()
> ```

Les sommets après l'étape de limite sont désormais récupérés de manière non optimale en raison de la TinkerPop modification 3.4.11. Pour éviter cela, vous pouvez modifier la requête en ajoutant l'étape barrier() à tout moment après `order().by()`. Par exemple :

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

TinkerPop La version 3.4.11 a été activée dans la version 1.0.5.0 du moteur Neptune.

Rubriques

- Configuration Heartbeat pour Neptune Serverless
- Structurer les requêtes upsert pour tirer parti du moteur DFE
- Testez le code Gremlin dans le contexte dans lequel vous allez le déployer
- Création d'écritures Gremlin multithreads efficaces
- Élagage des enregistrements avec la propriété de date/heure de création
- Utilisation de la méthode datetime () pour Groovy Time Data
- Utilisation de la date et de l'heure natives pour les données temporelles GLV

Configuration Heartbeat pour Neptune Serverless

Lorsque vous utilisez des WebSocket clients G705 avec Neptune Serverless, vous devez configurer l'intervalle de ping du client de manière appropriée afin de maintenir des connexions stables pendant les événements de dimensionnement. Le client G705 utilise des WebSocket connexions et envoie des pings périodiques pour vérifier que la connexion est active. Le client attend une réponse du serveur dans le délai imparti pour le ping. Si le serveur ne répond pas, le client ferme automatiquement la connexion.

Pour les instances provisionnées par Neptune, nous recommandons de définir l'intervalle ping sur 5 secondes. Pour les clusters Neptune Serverless, nous recommandons de définir l'intervalle ping sur au moins 20 secondes afin de tenir compte des retards potentiels lors des opérations de

dimensionnement. Ce paramètre contrôle la durée pendant laquelle le client attend entre deux écritures sur le serveur avant d'envoyer un ping pour vérifier que la connexion est toujours active.

La configuration de ce paramètre varie en fonction de l'implémentation du client :

Configuration du client Java

Pour le client Java TinkerPop Gremlin, configurez le `keepAliveInterval` paramètre :

```
Cluster.Builder builder = Cluster.build()
    .addContactPoint(endpoint)
    .keepAliveInterval(20000); // Configure ping interval in milliseconds
```

Pour plus de détails sur la configuration du pilote Java, consultez la TinkerPop documentation Java.

Configuration du client Go

Pour le client Gremlin Go, configurez le `KeepAliveInterval` paramètre :

```
rc, err := driver.NewDriverRemoteConnection(endpoint,
    func(settings *driver.DriverRemoteConnectionSettings) {
        settings.TraversalSource = "g"
        settings.AuthInfo = auth
        settings.KeepAliveInterval = 20 * time.Second // Configure ping interval
        ...
    })
```

Pour plus de détails sur la configuration du pilote Go, consultez la TinkerPop documentation Go.

JavaScript/Node.js Configuration du client

Pour le client JavaScript/Node.js Gremlin, configurez le `pingInterval` paramètre :

```
const gremlin = require('gremlin');
const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;

const connection = new DriverRemoteConnection(endpoint, {
    traversalSource: 'g',
    pingInterval: 20000  // Configure ping interval in milliseconds
});
```

Pour plus de détails sur la configuration du JavaScript pilote, reportez-vous à la JavaScript TinkerPop documentation.

Configuration du client Python

Pour le client Python Gremlin, l'intervalle ping est généralement géré au niveau de la couche de transport. Consultez la documentation spécifique de mise en œuvre du transport pour connaître les options de configuration :

```
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection

g = traversal().with_remote(
    DriverRemoteConnection('wss://your-neptune-endpoint:your-neptune-port/gremlin','g',
        transport_factory=lambda: AiohttpTransport(read_timeout=60,
                                                   write_timeout=20,
                                                   heartbeat=20, // Configure
 heartbeat
                                                   call_from_event_loop=True,
                                                   max_content_length=100*1024*1024,

 ssl_options=ssl.create_default_context(Purpose.CLIENT_AUTH))))
```

Pour plus de détails sur la configuration du pilote Python, consultez la TinkerPop documentation Python.

Cette configuration garantit à votre client la stabilité de la connexion lors des événements de dimensionnement Neptune Serverless, en évitant les fermetures de connexion inutiles et en améliorant la fiabilité des applications.

Structurer les requêtes upsert pour tirer parti du moteur DFE

Réaliser des upserts efficaces avec les étapes Gremlin MergeV () et MergeE () explique comment structurer les requêtes upsert pour utiliser le moteur DFE le plus efficacement possible.

Testez le code Gremlin dans le contexte dans lequel vous allez le déployer

Gremlin propose aux clients de multiples moyens de soumettre des requêtes au serveur. Ces modes de soumission diffèrent quant à la manière dont les requêtes sont évaluées et au comportement des transactions. Ces différences peuvent entraîner des résultats inattendus si vous développez dans un mode et déployez dans un autre.

Mode script (soumission basée sur des chaînes)

En mode script, le client envoie l'intégralité de la requête sous forme de chaîne de texte au serveur. Le serveur évalue la chaîne complète, y compris les étapes du terminal, et transmet les résultats au client. Les outils et méthodes suivants utilisent le mode script :

- Console Gremlin — lors de l'utilisation de la commande `:remote`
- Carnet graphique Neptune
- HTTP avec AWS SDK ou CLI AWS
- Soumission de chaînes de requête via des TinkerPop pilotes (par exemple, `client.submit("g.V().count()")` en utilisant Java)

En mode script, le client envoie une requête comme celle-ci au serveur sous forme de chaîne complète :

```
// Script mode - the entire string, including .next(), is sent to the server
// V('non-existing-id') yields nothing because no vertex with that ID exists
Cluster cluster = Cluster.build().addContactPoint("your-neptune-endpoint")
                         .port(8182).enableSsl(true).create();
Client client = cluster.connect();
client.submit("g.V('existing-id').addV('person').V('non-existing-id').next()");
```

Le serveur évalue la requête complète, y compris `.next()`. S'`.next()`il ne trouve aucun résultat, le serveur déclenche un `NoSuchElementException` et la transaction est annulée.

Mode bytecode (objets de traversée GLV)

En mode bytecode, le client crée un objet de traversée localement à l'aide d'une variante du langage G705 (GLV). Le pilote sérialise les étapes de traversée sous forme de bytecode et les envoie au serveur. Des étapes terminales telles que l'`.next()`exécution `.toList()` ou l'exécution côté client pour itérer les résultats renvoyés par le serveur. Vous pouvez utiliser le mode bytecode avec Java, Python, Go JavaScript, .NET et d'autres TinkerPop-compliant pilotes tiers adaptés à la version du moteur Neptune utilisée.

En mode bytecode, la même requête ressemble à ceci :

```
// Bytecode mode - the driver sends the traversal steps as bytecode
// .next() executes on the client to iterate results
```

```
Cluster cluster = Cluster.build().addContactPoint("your-neptune-endpoint")
                         .port(8182).enableSsl(true).create();
GraphTraversalSource g = traversal().withRemote(
    DriverRemoteConnection.using(cluster));
g.V("existing-id").addV("person").V("non-existing-id").next();
```

Le pilote envoie uniquement les étapes de traversée (`g.V().addV().V()`) sous forme de bytecode. Le serveur évalue correctement la traversée, valide la transaction et renvoie le jeu de résultats. Le client appelle ensuite `.next()` localement pour lire le jeu de résultats. Si le jeu de résultats est vide, le client déclenche un`NoSuchElementException`, mais la transaction a déjà été validée sur le serveur.

Différences de comportement des transactions

La différence essentielle entre ces modes réside dans la façon dont les étapes du terminal affectent les transactions :

- Mode script : le serveur évalue les étapes du terminal. Si une étape du terminal `.next()` échoue parce que le jeu de résultats est vide, le serveur considère la requête comme ayant échoué et annule la transaction. Le serveur ne conserve aucune mutation dans le même parcours (par exemple`addV()`).
- Mode bytecode : le client évalue les étapes du terminal. Le serveur évalue uniquement les étapes de traversée, valide correctement la transaction et renvoie les résultats. Si le client appelle `.next()` ensuite un jeu de résultats vide, il en `NoSuchElementException` résulte une erreur uniquement côté client. La transaction a déjà été validée, de sorte que le serveur conserve toutes les mutations.

Étapes du terminal

Dans G705, les étapes terminales sont les étapes qui entraînent la soumission d'une traversée à Neptune pour évaluation. En mode bytecode, l'étape du terminal déclenche la sérialisation et l'envoi de la traversée par le pilote. En mode script, l'étape du terminal fait partie de la chaîne de requête évaluée sur le serveur. Les étapes du terminal sont les suivantes :

- `hasNext()`— Renvoie `true` si des résultats sont disponibles, dans le `false` cas contraire.
- `next()`— Renvoie le résultat suivant. Lance `NoSuchElementException` si aucun résultat n'existe.
- `next(n)`— Renvoie les *n* résultats suivants sous forme de liste.

- `toList()`— Renvoie tous les résultats sous forme de liste. Renvoie une liste vide s'il n'existe aucun résultat.
- `toSet()`— Renvoie tous les résultats sous forme d'ensemble. Renvoie un ensemble vide s'il n'existe aucun résultat.
- `iterate()`— Réitère tous les résultats sans les renvoyer. Utilisez-le pour les mutations pour lesquelles vous n'avez pas besoin de la valeur de retour.

> **Note**
>
> Les GLV de langues individuelles peuvent fournir des étapes terminales supplémentaires spécifiques à leur mise en œuvre. Consultez les pages spécifiques à la langue pour plus de détails.

Si vous développez et testez votre code dans un contexte, vous pouvez rencontrer des problèmes. Par exemple, la console Gremlin envoie des requêtes sous forme de scripts. Votre code peut se comporter différemment en production si vous le déployez dans un contexte différent, par exemple via le pilote Java utilisant du bytecode.

> **Important**
>
> Assurez-vous de tester le code G705 en utilisant le même mode de soumission que celui dans lequel il sera déployé, afin d'éviter tout comportement de transaction inattendu.

Création d'écritures Gremlin multithreads efficaces

Quelques recommandations sont à observer pour le chargement multithread de données dans Neptune avec Gremlin.

Dans la mesure du possible, attribuez à chaque thread un ensemble de sommets ou d'arcs à insérer ou modifier qui ne se chevauchent pas. Par exemple, le thread 1 concerne la plage d'ID allant de 1 à 50 000, le thread 2 concerne la plage d'ID allant de 50 001 à 100 000, et ainsi de suite. Cela réduit le risque de générer une exception `ConcurrentModificationException`. Par mesure de sécurité, placez un bloc `try/catch` autour de toutes les écritures. En cas d'échec, vous pouvez effectuer une nouvelle tentative après un court délai. Pour obtenir des conseils sur le développement

d'une stratégie de nouvelle tentative avec un recul et une instabilité exponentiels, reportez-vous à la section. the section called "Exceptions et nouvelles tentatives"

Le traitement par lots de 50 à 100 écritures (sommets ou arcs) donne généralement de bons résultats. Si, pour chaque sommet, un grand nombre de propriétés sont ajoutées, il est préférable de tendre vers 50 plutôt que 100. Il est utile de réaliser des expérimentations. Ainsi, pour les écritures traitées par lots, vous pouvez utiliser un code similaire à ce qui suit :

```
g.addV('test').property(id,'1').as('a').
  addV('test').property(id,'2').
  addE('friend').to('a').
```

Il est ensuite repris dans chaque opération de traitement par lots.

Il est nettement plus efficace d'utiliser des lots que d'ajouter un sommet ou un arc au serveur pour chaque boucle Gremlin.

Si vous utilisez un client GLV (Gremlin Language Variant), vous pouvez créer un lot par programmation en commençant par la création d'un parcours. Ensuite, effectuez des ajouts et, enfin, des itérations, par exemple :

```
  t.addV('test').property(id,'1').as('a')
  t.addV('test').property(id,'2')
  t.addE('friend').to('a')
  t.iterate()
```

Il est préférable d'utiliser le client GLV (Gremlin Language Variant) si possible. Mais vous pouvez faire quelque chose de similaire avec un client qui soumet des requêtes sous forme de chaînes de texte qui sont concaténées pour former un lot.

Si vous utilisez l'une des bibliothèques de client Gremlin plutôt que le protocole HTTP de base pour les requêtes, les threads doivent tous partager le même client, cluster ou groupe de connexion. Vous pouvez être amené à ajuster les paramètres pour bénéficier d'un débit optimal, des paramètres tels que la taille du groupe de connexion et le nombre de threads de travail qu'utilise le client Gremlin.

Élagage des enregistrements avec la propriété de date/heure de création

Vous pouvez élaguer les enregistrements obsolètes en stockant la date/heure de création en tant que propriété sur les sommets et en les supprimant périodiquement.

Si vous devez stocker des données pour une durée de vie spécifique, puis les supprimer du graphe (durée de vie du sommet), vous pouvez stocker une propriété d'horodatage lors de la création du sommet. Vous pouvez ensuite exécuter périodiquement une requête `drop()` pour tous les sommets qui ont été créés avant une certaine date/heure, par exemple :

```
g.V().has(“timestamp”, lt(datetime('2018-10-11')))
```

Utilisation de la méthode **`datetime ()`** pour Groovy Time Data

Neptune fournit la méthode `datetime` afin de spécifier des dates et heures pour les requêtes envoyées dans la variante Gremlin Groovy. Cela inclut la console Gremlin, les chaînes de texte utilisant l'API REST HTTP, et toute autre sérialisation qui utilise Groovy.

> **Important**
>
> Ceci s'applique uniquement aux méthodes dans lesquelles vous envoyez la requête Gremlin en tant que chaîne de texte. Si vous utilisez une variante du langage Gremlin (Gremlin Language Variant), vous devez utiliser les classes et fonctions de dates natives pour le langage. Pour plus d'informations, consultez la section suivante, the section called “Date et heure natives”.
>
> Commencer par TinkerPop `3.5.2` (introduit dans la version 1.1.1.0 du moteur Neptune) fait partie intégrante de. `datetime` TinkerPop

Vous pouvez utiliser la méthode `datetime` pour stocker et comparer des dates :

```
g.V('3').property('date',datetime('2001-02-08'))
```

```
g.V().has('date',gt(datetime('2000-01-01')))
```

Utilisation de la date et de l'heure natives pour les données temporelles GLV

Si vous utilisez une variante du langage Gremlin (GLV), vous devez utiliser les classes et fonctions de date et d'heure natives fournies par le langage de programmation pour les données temporelles Gremlin.

Les TinkerPop bibliothèques officielles sont toutes des bibliothèques de variantes de langage Gremlin.

- Go
- Java
- JavaScript
- .NET
- Python

> **Important**
>
> Cette page s'applique uniquement aux bibliothèques GLV (Grupal Language Variant). Si vous utilisez une méthode dans laquelle vous envoyez la requête G705 sous forme de chaîne de texte, vous devez utiliser la fonction datetime () de G705. Cela inclut la console G705, les chaînes de texte utilisant l'API HTTP REST ou la soumission directe de chaînes G705 via les pilotes.

Go

Voici un exemple partiel dans Go qui crée une propriété unique nommée « date » pour le sommet avec un ID de « 3 ». Il définit la valeur comme étant une date générée à l'aide de la fonction Go Time.now ().

```
import ( "time" )

g.V('3').property('date', time.Now()).next();
```

Pour un exemple complet de connexion à Neptune à l'aide de Go, voir Utilisation d'un client Go pour se connecter à une instance de base de données Neptune.

Java

Voici un exemple partiel en Java qui crée une propriété unique nommée « `date` » pour le sommet avec l'ID « `3` ». Cet exemple définit la valeur sur une date générée à l'aide du constructeur `Date()` Java.

```
import java.util.date

g.V('3').property('date', new Date()).next();
```

Pour un exemple complet de connexion à Neptune à l'aide de Java, consultez. Utilisation du client Java pour se connecter à une instance de base de données Neptune

Node.js (JavaScript)

Voici un exemple partiel de création JavaScript d'une propriété unique nommée « `date` » pour le sommet avec un ID de « `3` ». Il définit la valeur comme étant une date générée à l'aide du Node.js `Date()` constructeur.

```
g.V('3').property('date', new Date()).next()
```

Pour un exemple complet de connexion à Neptune à l'aide Node.js de. Utilisation Node.js pour se connecter à une instance de base de données Neptune

.NET (C#)

Voici un exemple partiel en C# qui crée une propriété unique nommée « `date` » pour le sommet avec l'ID « `3` ». Cet exemple définit la valeur sur une date générée à l'aide de la propriété .NET *`DateTime.UtcNow`*.

```
Using System;

g.V('3').property('date', DateTime.UtcNow).next()
```

Pour un exemple complet de connexion à Neptune à l'aide de C#, consultez. Utilisation de .NET pour se connecter à une instance de base de données Neptune

Python

Voici un exemple partiel en Python qui crée une propriété unique nommée « `date` » pour le sommet avec l'ID « `3` ». Cet exemple définit la valeur sur une date générée à l'aide de la méthode `datetime.now()` Python.

```
import datetime

g.V('3').property('date',datetime.datetime.now()).next()
```

Pour un exemple complet de connexion à Neptune à l'aide de Python, consultez. Utilisation de Python pour se connecter à une instance de base de données Neptune

Bonnes pratiques pour l'utilisation du client Java Gremlin avec Neptune

Suivez ces recommandations lors de l'utilisation du client Java Gremlin avec Neptune. Ces bonnes pratiques vous aident à optimiser les performances, à gérer efficacement les connexions et à éviter les pièges courants lorsque vous utilisez le pilote Java.

Pour plus d'informations sur la configuration des intervalles de pulsation pour Neptune Serverless, consultez. Configuration Heartbeat pour Neptune Serverless

Rubriques

- Réutilisation de l'objet client dans plusieurs threads
- Création d'objets client Java Gremlin distincts pour les points de terminaison en lecture et en écriture
- Ajout de plusieurs points de terminaison de réplica en lecture à un groupe de connexions Java Gremlin
- Fermeture du client pour éviter de limiter les connexions
- Création d'une connexion après un basculement
- Définition de maxInProcessPerConnection et maxSimultaneousUsagePerConnection sur la même valeur
- Envoi de requêtes au serveur sous la forme de bytecode et non de chaînes
- Consommez toujours complètement l'itérateur ResultSet ou l'itérateur renvoyé par une requête
- Ajout en bloc de sommets et d'arêtes dans des lots
- Désactivation de la mise en cache du DNS dans la machine virtuelle Java
- Définition facultative de délais d'expiration au niveau de chaque requête
- Résolution des problèmes de java.util.concurrent.TimeoutException

Réutilisation de l'objet client dans plusieurs threads

Réutilisez le même objet client (ou `GraphTraversalSource`) dans plusieurs threads. Autrement dit, créez une instance partagée d'une `org.apache.tinkerpop.gremlin.driver.Client` classe

dans votre application plutôt que de le faire dans chaque thread. L'objet `Client` est thread-safe, et la surcharge entraînée par l'initialisation est considérable.

Cela s'applique également à `GraphTraversalSource`, qui crée un objet `Client` en interne. Par exemple, le code suivant entraîne l'instanciation d'un nouvel objet `Client` :

```
import static
 org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource.traversal;

  /////

GraphTraversalSource traversal = traversal()
                                    .withRemote(DriverRemoteConnection.using(cluster));
```

Création d'objets client Java Gremlin distincts pour les points de terminaison en lecture et en écriture

Vous pouvez augmenter les performances en effectuant uniquement les écritures sur le point de terminaison enregistreur et la lecture à partir d'un ou plusieurs points de terminaison en lecture seule.

```
Client readerClient = Cluster.build("https://reader-endpoint")
          ...
          .connect()

Client writerClient = Cluster.build("https://writer-endpoint")
          ...
          .connect()
```

Ajout de plusieurs points de terminaison de réplica en lecture à un groupe de connexions Java Gremlin

Lorsque vous créez un objet `Cluster` Java Gremlin, vous pouvez utiliser la méthode `.addContactPoint()` pour ajouter plusieurs instances de réplicas en lecture aux points de contact du groupe de connexions.

```
Cluster.Builder readerBuilder = Cluster.build()
          .port(8182)
          .minConnectionPoolSize(…)
          .maxConnectionPoolSize(…)
          ……
```

```
            .addContactPoint("reader-endpoint-1")
            .addContactPoint("reader-endpoint-2")
```

Fermeture du client pour éviter de limiter les connexions

Il est important de fermer le client lorsque vous avez terminé de l'utiliser pour vous assurer que les WebSocket connexions sont fermées par le serveur et que toutes les ressources associées aux connexions sont libérées. Cela se produit automatiquement si vous fermez le cluster à l'aide de `Cluster.close( )`, car `client.close( )` est ensuite appelé en interne.

Si le client n'est pas fermé correctement, Neptune met fin à toutes les WebSocket connexions inactives au bout de 20 à 25 minutes. Toutefois, si vous ne fermez pas explicitement les WebSocket connexions lorsque vous en avez terminé et que le nombre de connexions en direct atteint la limite de connexions WebSocket simultanées, les connexions supplémentaires sont alors refusées avec un code 429 d'erreur HTTP. À ce stade, vous devez redémarrer l'instance Neptune pour fermer les connexions.

Le conseil avisant d'appeler `cluster.close()` ne s'applique pas aux fonctions AWS Lambda Java. Consultez Gestion des WebSocket connexions G705 dans les fonctions AWS Lambda pour plus de détails.

Création d'une connexion après un basculement

En cas de basculement, le pilote Gremlin peut continuer à se connecter à l'ancien enregistreur, car le nom DNS du cluster a été résolu en une adresse IP. Si cela se produit, vous pouvez créer un nouvel objet `Client` après le basculement.

Définition de **`maxInProcessPerConnection`** et **`maxSimultaneousUsagePerConnection`** sur la même valeur

Les paramètres `maxInProcessPerConnection` et les `maxSimultaneousUsagePerConnection` paramètres sont liés au nombre maximum de requêtes simultanées que vous pouvez soumettre sur une seule WebSocket connexion. En interne, ces paramètres sont liés entre eux et la modification de l'un sans modifier l'autre peut entraîner la réception d'un délai d'expiration lors d'une tentative d'extraction d'une connexion à partir du groupe de connexions du client.

Nous vous recommandons de conserver la valeur minimale par défaut en cours et la valeur d'utilisation simultanée, et de définir `maxInProcessPerConnection` et `maxSimultaneousUsagePerConnection` sur la même valeur.

La valeur sur laquelle définir ces paramètres dépend de la complexité de la requête et du modèle de données. Un cas d'utilisation dans lequel la requête renvoie un grand nombre de données nécessite plus de bande passante de connexion par requête. Par conséquent, les paramètres doivent être définis sur des valeurs plus faibles et `maxConnectionPoolSize` sur une valeur plus élevée.

En revanche, dans le cas où la requête renvoie une plus petite quantité de données, `maxInProcessPerConnection` et `maxSimultaneousUsagePerConnection` doivent être définis sur une valeur plus élevée que `maxConnectionPoolSize`.

Envoi de requêtes au serveur sous la forme de bytecode et non de chaînes

L'utilisation de bytecode plutôt que d'une chaîne lors de la soumission de requêtes présente plusieurs avantages :

- Interception plus rapide d'une syntaxe de requête non valide : L'utilisation de la variante bytecode permet de détecter une syntaxe de requête non valide lors de l'étape de compilation. Si vous utilisez la variante basée sur une chaîne, vous ne pouvez pas identifier la syntaxe non valide tant que la requête n'est pas soumise au serveur et qu'une erreur n'est pas renvoyée.
- Évitez de réduire les performances basées sur des chaînes : toute soumission de requête basée sur des chaînes, que vous utilisiez HTTP WebSockets ou HTTP, entraîne le détachement d'un sommet, ce qui implique que l'objet Vertex comprend l'ID, l'étiquette et toutes les propriétés associées au sommet (voir Propriétés des éléments).

  Dans les cas où les propriétés ne sont pas obligatoires, cela peut entraîner des calculs inutiles sur le serveur. Par exemple, si le client souhaite obtenir le vertex portant l'ID « hakuna#1 » à l'aide de la requête `g.V("hakuna#1")`. Si la requête est envoyée sous la forme d'une soumission basée sur une chaîne, le serveur consacre du temps à la récupération de l'ID, de l'étiquette et de toutes les propriétés de ce vertex. Si la requête est envoyée sous la forme d'une soumission de bytecode, le serveur consacre du temps à la récupération de l'ID et de l'étiquette du vertex uniquement.

En d'autres termes, au lieu de soumettre une requête comme suit :

```
final Cluster cluster = Cluster.build("localhost")
                              .port(8182)
                              .maxInProcessPerConnection(32)
                              .maxSimultaneousUsagePerConnection(32)
                              .serializer(Serializers.GRAPHBINARY_V1D0)
                              .create();
```

```
 try {
     final Client client = cluster.connect();
     List<Result> results =
client.submit("g.V().has('name','pumba').out('friendOf').id()").all().get();
     System.out.println(verticesWithNamePumba);
 } finally {
     cluster.close();
 }
```

Soumettez la requête en bytecode comme suit :

```
 final Cluster cluster = Cluster.build("localhost")
                               .port(8182)
                               .maxInProcessPerConnection(32)
                               .maxSimultaneousUsagePerConnection(32)
                               .serializer(Serializers.GRAPHBINARY_V1D0)
                               .create();

 try {
     final GraphTraversalSource g =
traversal().withRemote(DriverRemoteConnection.using(cluster));
     List<Object> verticesWithNamePumba = g.V().has("name",
"pumba").out("friendOf").id().toList();
     System.out.println(verticesWithNamePumba);
 } finally {
     cluster.close();
 }
```

Consommez toujours complètement l'itérateur ResultSet ou l'itérateur renvoyé par une requête

L'objet client doit toujours utiliser intégralement le `ResultSet` (dans le cas d'une soumission basée sur une chaîne) ou l'itérateur renvoyé par `GraphTraversal`. Si les résultats de la requête ne sont pas intégralement utilisés, le serveur les conserve et attend que le client finisse de les utiliser.

Si votre application a uniquement besoin d'une partie des résultats, vous pouvez utiliser une étape `limit(X)` avec votre requête pour limiter le nombre de résultats générés par le serveur.

Ajout en bloc de sommets et d'arêtes dans des lots

Chaque requête envoyée à la base de données Neptune s'exécute dans le cadre d'une seule transaction, sauf si vous utilisez une session. Autrement dit, si vous devez insérer un grand nombre de données à l'aide de requêtes Gremlin, leur traitement dans un lot contenant entre 50 et 100 éléments permet d'améliorer les performances en réduisant le nombre de transactions créées pour la charge.

Par exemple, l'ajout de 5 vertex à la base de données se présente comme suit :

```
// Create a GraphTraversalSource for the remote connection
final GraphTraversalSource g =
 traversal().withRemote(DriverRemoteConnection.using(cluster));
// Add 5 vertices in a single query
g.addV("Person").property(T.id, "P1")
 .addV("Person").property(T.id, "P2")
 .addV("Person").property(T.id, "P3")
 .addV("Person").property(T.id, "P4")
 .addV("Person").property(T.id, "P5").iterate();
```

De même, vous pouvez ajouter des arêtes par lots à l'aide de. `addE V()`À utiliser pour référencer des sommets existants en tant que source et cible de chaque arête :

```
// Add edges in a single batched query
g.V("P1").addE("knows").to(V("P2"))
 .V("P2").addE("knows").to(V("P3"))
 .V("P3").addE("knows").to(V("P4"))
 .V("P4").addE("knows").to(V("P5")).iterate();
```

Vous pouvez également combiner la création de sommets et d'arêtes en un seul lot. `as()`À utiliser pour étiqueter les sommets nouvellement créés afin de pouvoir les référencer lorsque vous ajoutez des arêtes dans le même parcours :

```
// Add vertices and edges together in a single query
g.addV("Person").property(T.id, "P1").as("p1")
 .addV("Person").property(T.id, "P2").as("p2")
 .addV("Person").property(T.id, "P3").as("p3")
 .addE("knows").from("p1").to("p2")
 .addE("knows").from("p2").to("p3").iterate();
```

Désactivation de la mise en cache du DNS dans la machine virtuelle Java

Dans un environnement où vous souhaitez équilibrer la charge des requêtes entre plusieurs répliques de lecture, vous devez désactiver la mise en cache DNS dans la machine virtuelle Java (JVM) et fournir le point de terminaison du lecteur Neptune lors de la création de l'objet Cluster. La désactivation du cache du DNS dans la JVM garantit que le DNS est à nouveau résolu pour chaque nouvelle connexion, de sorte que les demandes soient réparties entre tous les réplicas en lecture. Pour ce faire, ajoutez la ligne suivante dans le code d'initialisation de votre application :

```
java.security.Security.setProperty("networkaddress.cache.ttl", "0");
```

Cependant, une solution plus complète et plus robuste pour l'équilibrage de charge est fournie par le code client Java Amazon Gremlin sur. GitHub Le client Amazon Java Gremlin connaît la topologie de votre cluster et répartit équitablement les connexions et les demandes entre un ensemble d'instances du cluster Neptune. Consultez ce billet de blog pour découvrir un exemple de fonction Lambda Java qui utilise ce client.

Définition facultative de délais d'expiration au niveau de chaque requête

Neptune offre la possibilité de définir un délai d'expiration pour vos requêtes en utilisant l'option de groupe de paramètres `neptune_query_timeout` (consultez Parameters). Vous pouvez également annuler le délai d'attente global, avec un code comme celui-ci :

> **Note**
>
> Les paramètres de délai d'attente des requêtes s'appliquent uniquement à l'évaluation des requêtes. Les opérations de contrôle des transactions basées sur le bytecode, telles que `tx().commit()` et ne `tx().rollback()` sont pas soumises à des délais d'attente pour les requêtes. Pour de plus amples informations, veuillez consulter Comportement du délai d'expiration pour la validation et l'annulation du bytecode.

```
final Cluster cluster = Cluster.build("localhost")
                               .port(8182)
                               .maxInProcessPerConnection(32)
                               .maxSimultaneousUsagePerConnection(32)
                               .serializer(Serializers.GRAPHBINARY_V1D0)
                               .create();
```

```
 try {
     final GraphTraversalSource g =
traversal().withRemote(DriverRemoteConnection.using(cluster));
     List<Object> verticesWithNamePumba = g.with(ARGS_EVAL_TIMEOUT,
500L).V().has("name", "pumba").out("friendOf").id().toList();
     System.out.println(verticesWithNamePumba);
 } finally {
     cluster.close();
 }
```

Sinon, le code se présente comme suit pour la soumission d'une requête basée sur une chaîne :

```
RequestOptions options = RequestOptions.build().timeout(500).create();
List<Result> result = client.submit("g.V()", options).all().get();
```

Si une requête dépasse le délai d'expiration par requête, Neptune y met fin. L'opportunité de réessayer une requête dont le délai est expiré dépend de la nature de l'échec et de votre charge de travail. Pour de plus amples informations, consultez the section called “Exceptions et nouvelles tentatives”.

> **Note**
>
> Il est possible que vous encouriez des coûts inattendus si vous définissez une valeur trop élevée pour le délai d'expiration des requêtes, en particulier sur une instance sans serveur. En l'absence d'un délai raisonnable d'expiration des requêtes, la requête risque de s'exécuter bien plus longtemps que prévu, entraînant ainsi des coûts que vous n'aviez pas anticipés. Cela est particulièrement vrai pour une instance sans serveur, qui pourrait passer à un type d'instance volumineux et coûteux lors de l'exécution de la requête.
> Pour éviter des dépenses imprévues de ce type, utilisez une valeur de délai d'expiration qui s'adapte au temps d'éxécution prévu et qui n'implique que l'expiration des exécutions dont le délai est étonnamment long.
> À partir de la version 1.3.2.0 du moteur Neptune, Neptune prend en charge un nouveau paramètre neptune_lab_mode sous la forme. `StrictTimeoutValidation` Lorsque ce paramètre a une valeur de`Enabled`, une valeur de délai par requête spécifiée en tant qu'option de demande ou indication de requête ne peut pas dépasser la valeur définie globalement dans le groupe de paramètres. Dans ce cas, Neptune lancera. `InvalidParameterException`
> Ce paramètre peut être confirmé dans une réponse sur le point de terminaison « / status » lorsque la valeur est`Disabled`. Dans la version du moteur`1.3.2.0`, la valeur

par défaut de ce paramètre est`Disabled`. À partir de la version du moteur`1.4.0.0`, le `StrictTimeoutValidation` paramètre est défini `Enabled` par défaut.
Pour plus d'informations sur la façon dont la priorité du délai d'expiration est déterminée lorsque plusieurs paramètres de délai d'expiration sont configurés, consultez la documentation du paramètre neptune_query_timeout.

Résolution des problèmes de `java.util.concurrent.TimeoutException`

Le client Java Gremlin envoie au client lui-même un message indiquant `java.util.concurrent.TimeoutException` quand une requête Garmin expire en attendant qu'un emplacement soit disponible dans l'une des WebSocket connexions. Ce délai d'expiration est contrôlé par le paramètre configurable `maxWaitForConnection` côté client.

> **Note**
>
> Comme les demandes qui expirent sur le client ne sont jamais envoyées au serveur, elles ne sont reflétées dans aucune des métriques capturées sur le serveur, telles que `GremlinRequestsPerSec`.

Ce type de délai d'expiration est généralement causé de l'une des deux manières suivantes :

- Le serveur a atteint sa capacité maximale. Dans ce cas, la file d'attente du serveur se remplit, ce que vous pouvez détecter en surveillant la MainRequestQueuePendingRequests CloudWatch métrique. Le nombre de requêtes parallèles que le serveur peut traiter dépend de la taille de son instance.

  Si la métrique `MainRequestQueuePendingRequests` n'indique pas une accumulation des demandes en attente sur le serveur, le serveur peut traiter un plus grand nombre de demandes, et l'expiration est donc due à une limitation côté client.
- Limitation des demandes côté client. Ce problème peut généralement être résolu en modifiant les paramètres de configuration du client.

  Le nombre maximal de demandes parallèles que le client peut envoyer peut être estimé approximativement comme suit :

```
maxParallelQueries = maxConnectionPoolSize * Max( maxSimultaneousUsagePerConnection,
 maxInProcessPerConnection )
```

L'envoi d'un nombre de demandes supérieur à `maxParallelQueries` au client entraîne des exceptions `java.util.concurrent.TimeoutException`. Vous pouvez généralement résoudre ce problème de plusieurs manières :

- Augmentez le délai d'expiration de la connexion. Si la latence n'est pas cruciale pour votre application, augmentez la valeur du paramètre `maxWaitForConnection` du client. Le client attendra plus longtemps avant l'expiration du délai, ce qui contribue à accroître la latence.
- Augmentez le nombre maximum de demandes par connexion. Cela permet d'envoyer un plus grand nombre de demandes via la même WebSocket connexion. Pour ce faire, augmentez la valeur des paramètres `maxSimultaneousUsagePerConnection` et `maxInProcessPerConnection` du client. Ces paramètres doivent généralement avoir la même valeur.
- Augmentez le nombre de connexions dans le groupe de connexions. Pour ce faire, augmentez la valeur du paramètre `maxConnectionPoolSize` du client. Le coût est une augmentation de la consommation de ressources, car chaque connexion utilise de la mémoire et un descripteur de fichier du système d'exploitation, et nécessite un protocole SSL et WebSocket une poignée de main lors de l'initialisation.

Bonnes pratiques Neptune avec openCypher et Bolt

Suivez ces bonnes pratiques lorsque vous utilisez le langage de requête openCypher et le protocole Bolt avec Neptune. Pour plus d'informations sur l'utilisation d'openCypher dans Neptune, consultez Accès au graphe Neptune avec openCypher.

Rubriques

- Création d'une connexion après un basculement
- Gestion des connexions pour les applications de longue durée
- Gestion des connexions pour AWS Lambda
- Préférer les arêtes dirigées aux arêtes bidirectionnelles dans les requêtes
- Neptune ne prend pas en charge plusieurs requêtes simultanées dans une transaction
- Fermer les objets Driver lorsque vous avez terminé
- Utilisation des modes de transaction explicites pour la lecture et l'écriture

- Logique des nouvelles tentatives pour les exceptions
- Définissez plusieurs propriétés à la fois à l'aide d'une seule clause SET
- Utilisation des requêtes paramétrées
- Utilisez des cartes aplaties au lieu de cartes imbriquées dans la clause UNWIND
- Placez des nœuds plus restrictifs sur le côté gauche dans les expressions VLP (Variable-Length Path)
- Évitez les vérifications redondantes des étiquettes des nœuds en utilisant des noms de relations granulaires
- Spécifiez les étiquettes de bord dans la mesure du possible
- Évitez d'utiliser la clause WITH dans la mesure du possible
- Placez les filtres restrictifs le plus tôt possible dans la requête
- Vérifiez explicitement si les propriétés existent
- N'utilisez pas de chemin nommé (sauf si cela est obligatoire)
- Évitez COLLECT (DISTINCT ())
- Préférez la fonction de propriétés à la recherche de propriétés individuelle lors de la récupération de toutes les valeurs de propriété
- Effectuer des calculs statiques en dehors de la requête
- Entrées par lots utilisant UNWIND au lieu d'instructions individuelles
- Préférez utiliser la personnalisation IDs pour le nœud ou la relation
- Évitez de faire des calculs ~id dans la requête
- Mise à jour/fusion de plusieurs nœuds

Création d'une connexion après un basculement

En cas de basculement, le pilote Bolt peut continuer à se connecter à l'ancienne instance d'enregistreur plutôt qu'à la nouvelle instance active, car le nom DNS a été résolu en une adresse IP spécifique.

Pour éviter cela, fermez puis reconnectez l'objet `Driver` après un basculement.

Gestion des connexions pour les applications de longue durée

Lorsque vous créez des applications à longue durée de vie, telles que celles qui s'exécutent dans des conteneurs ou sur des instances Amazon EC2, instanciez un objet `Driver` une fois, puis réutilisez-

le pendant toute la durée de vie de l'application. L'objet `Driver` est thread-safe, et la surcharge entraînée par l'initialisation est considérable.

Gestion des connexions pour AWS Lambda

Les pilotes Bolt ne sont pas recommandés pour une utilisation dans AWS Lambda les fonctions, en raison de leur surcharge de connexion et des exigences de gestion. Utilisez plutôt le point de terminaison HTTPS.

Préférer les arêtes dirigées aux arêtes bidirectionnelles dans les requêtes

Lorsque Neptune optimise les requêtes, les arêtes bidirectionnelles compliquent la création de plans de requêtes optimaux. Les plans sous-optimaux obligent le moteur à effectuer des tâches superflues, ce qui entraîne une baisse des performances.

Par conséquent, utilisez des arêtes dirigées plutôt que des arêtes bidirectionnelles dans la mesure du possible. Par exemple, utilisez :

```
MATCH p=(:airport {code: 'ANC'})-[:route]->(d) RETURN p)
```

au lieu de :

```
MATCH p=(:airport {code: 'ANC'})-[:route]-(d) RETURN p)
```

La plupart des modèles de données n'ont pas besoin de traverser les arêtes dans les deux sens. Les requêtes peuvent donc améliorer considérablement les performances avec l'utilisation d'arêtes dirigées.

Si le modèle de données nécessite de traverser des arêtes bidirectionnelles, faites du premier nœud (côté gauche) le nœud avec le filtrage le plus restrictif dans le modèle `MATCH`.

Prenons un exemple recherchant tous les itinéraires (`routes`) à destination et en provenance de l'aéroport `ANC`. Écrivez cette requête pour commencer à l'aéroport `ANC`, comme suit :

```
MATCH p=(src:airport {code: 'ANC'})-[:route]-(d) RETURN p
```

Le moteur peut ainsi effectuer un minimum d'efforts pour satisfaire la requête, car le nœud le plus restreint est placé en tant que premier nœud (côté gauche) dans le modèle. Puis, il peut optimiser la requête.

Cette approche est de loin préférable à un filtrage de l'aéroport ANC à la fin du modèle, comme ceci :

```
MATCH p=(d)-[:route]-(src:airport {code: 'ANC'}) RETURN p
```

Lorsque le nœud le plus restreint n'est pas placé en premier dans le modèle, le moteur doit effectuer des efforts supplémentaires, car il ne peut pas optimiser la requête et doit réaliser des recherches supplémentaires pour obtenir les résultats.

Neptune ne prend pas en charge plusieurs requêtes simultanées dans une transaction

Bien que le pilote Bolt lui-même autorise les requêtes simultanées dans une transaction, Neptune ne prend pas en charge plusieurs requêtes dans le cadre d'une transaction exécutée simultanément. Neptune exige plutôt que plusieurs requêtes d'une transaction soient exécutées de manière séquentielle et que les résultats de chaque requête soient entièrement utilisés avant le lancement de la prochaine requête.

L'exemple ci-dessous montre comment utiliser Bolt pour exécuter plusieurs requêtes de manière séquentielle dans une transaction, de sorte que les résultats de chacune soient entièrement consommés avant le début de la suivante :

```
final String query = "MATCH (n) RETURN n";

try (Driver driver = getDriver(HOST_BOLT, getDefaultConfig())) {
  try (Session session = driver.session(readSessionConfig)) {
    try (Transaction trx = session.beginTransaction()) {
      final Result res_1 = trx.run(query);
      Assert.assertEquals(10000, res_1.list().size());
      final Result res_2 = trx.run(query);
      Assert.assertEquals(10000, res_2.list().size());
    }
  }
}
```

Fermer les objets Driver lorsque vous avez terminé

Il est important de fermer le client lorsque vous n'en avez plus besoin afin de garantir que les connexions Bolt sont fermées par le serveur et que toutes les ressources associées aux connexions sont libérées. Cela se produit automatiquement si vous fermez le pilote à l'aide de `driver.close()`.

Si le pilote n'est pas correctement fermé, Neptune met fin à toutes les connexions Bolt inactives au bout de 20 minutes, ou au bout de 10 jours si vous utilisez l'authentification IAM.

Neptune prend en charge jusqu'à 1 000 connexions Bolt simultanées. Si vous ne fermez pas explicitement les connexions lorsque vous n'en avez plus besoin, et que le nombre de connexions en direct atteint cette limite de 1 000, toute nouvelle tentative de connexion échouera.

Utilisation des modes de transaction explicites pour la lecture et l'écriture

Lorsque vous utilisez des transactions avec Neptune et le pilote Bolt, il est préférable de définir explicitement le mode d'accès des transactions en lecture et en écriture avec les paramètres appropriés.

Transactions en lecture seule

Pour les transactions en lecture seule, si vous ne transmettez pas la configuration de mode d'accès appropriée lors de la création de la session, le niveau d'isolement par défaut est utilisé, à savoir l'isolement des requêtes de mutation. Par conséquent, il est important de définir le mode d'accès `read` de manière explicite pour les transactions en lecture seule.

Exemple de transaction de lecture avec validation automatique :

```
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.READ)
  .build();
Session session = driver.session(sessionConfig);
try {
  (Add your application code here)
} catch (final Exception e) {
  throw e;
} finally {
  driver.close()
}
```

Exemple de transaction de lecture :

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withDefaultAccessMode(AccessMode.READ)
```

```
  .build();
driver.session(sessionConfig).readTransaction(
  new TransactionWork<List<String>>() {
    @Override
    public List<String> execute(org.neo4j.driver.Transaction tx) {
      (Add your application code here)
    }
  }
);
```

Dans les deux cas, l'isolement SNAPSHOT est réalisé à l'aide de la sémantique des transactions en lecture seule de Neptune.

Comme les réplicas en lecture n'acceptent que les requêtes en lecture seule, toute requête soumise à un réplica en lecture s'exécute selon une sémantique d'isolement SNAPSHOT.

Il n'y a pas de lectures corrompues ni de lectures non reproductibles pour les transactions en lecture seule.

Transactions de mutation

Pour les requêtes de mutation, il existe trois mécanismes différents pour créer une transaction d'écriture, chacun étant illustré ci-dessous :

Exemple de transaction d'écriture implicite :

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();
driver.session(sessionConfig).writeTransaction(
  new TransactionWork<List<String>>() {
    @Override
    public List<String> execute(org.neo4j.driver.Transaction tx) {
      (Add your application code here)
    }
  }
);
```

Exemple de transaction d'écriture à validation automatique :

```
SessionConfig sessionConfig = SessionConfig
```

```
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.Write)
  .build();
Session session = driver.session(sessionConfig);
try {
  (Add your application code here)
} catch (final Exception e) {
    throw e;
} finally {
    driver.close()
}
```

Exemple de transaction d'écriture explicite :

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();
Transaction beginWriteTransaction = driver.session(sessionConfig).beginTransaction();
  (Add your application code here)
beginWriteTransaction.commit();
driver.close();
```

Niveaux d'isolement pour les transactions d'écriture

- Les lectures effectuées dans le cadre de requêtes de mutation sont exécutées dans le cadre de l'isolement des transactions `READ COMMITTED`.
- Il n'existe aucune lecture corrompue pour les lectures effectuées dans le cadre de requêtes de mutation.
- Les enregistrements et les plages d'enregistrements sont verrouillés lors de la lecture d'une requête de mutation.
- Lorsqu'une plage de l'index a été lue par une transaction de mutation, vous bénéficiez de la garantie que cette plage ne sera pas modifiée par des transactions simultanées jusqu'à la fin de la lecture.

Les requêtes de mutation ne sont pas thread-safe.

Pour les conflits, voir Résolution des conflits à l'aide de Lock-Wait délais.

Les requêtes de mutation ne font pas automatiquement l'objet de nouvelles tentatives en cas d'échec.

Logique des nouvelles tentatives pour les exceptions

Toutes les exceptions ne doivent pas être réessayées. Par exemple, les erreurs de syntaxe, les échecs d'authentification ou les requêtes non valides ne doivent pas déclencher une nouvelle tentative. Pour les exceptions qui autorisent une nouvelle tentative, utilisez une stratégie de réduction exponentielle avec instabilité afin d'augmenter progressivement le temps d'attente entre les tentatives et d'éviter les tempêtes de nouvelles tentatives synchronisées. Définissez un nombre maximum raisonnable de tentatives pour éviter les boucles infinies et l'épuisement des ressources. Pour obtenir des conseils détaillés sur le développement d'une stratégie de réessai pratique, voirthe section called "Exceptions et nouvelles tentatives".

Vous trouverez ci-dessous un exemple de schéma de réduction exponentielle et de nouvelles tentatives pour gérer les erreurs : `ConcurrentModificationException`

```
public static void main() {
  try (Driver driver = getDriver(HOST_BOLT, getDefaultConfig())) {
    retriableOperation(driver, "CREATE (n {prop:'1'})")
        .withRetries(5)
        .withExponentialBackoff(true)
        .maxWaitTimeInMilliSec(500)
        .call();
  }
}

protected RetryableWrapper retriableOperation(final Driver driver, final String query){
  return new RetryableWrapper<Void>() {
    @Override
    public Void submit() {
      log.info("Performing graph Operation in a retry manner......");
      try (Session session = driver.session(writeSessionConfig)) {
        try (Transaction trx =  session.beginTransaction()) {
            trx.run(query).consume();
            trx.commit();
        }
      }
      return null;
    }
```

```
    @Override
    public boolean isRetryable(Exception e) {
      if (isCME(e)) {
        log.debug("Retrying on exception.... {}", e);
        return true;
      }
      return false;
    }

    private boolean isCME(Exception ex) {
      return ex.getMessage().contains("Operation failed due to conflicting concurrent
 operations");
    }
  };
}



/**
 * Wrapper which can retry on certain condition. Client can retry operation using this
 class.
 */
@Log4j2
@Getter
public abstract class RetryableWrapper<T> {

  private long retries = 5;
  private long maxWaitTimeInSec = 1;
  private boolean exponentialBackoff = true;

  /**
   * Override the method with custom implementation, which will be called in retryable
 block.
   */
  public abstract T submit() throws Exception;

  /**
   * Override with custom logic, on which exception to retry with.
   */
  public abstract boolean isRetryable(final Exception e);

  /**
   * Define the number of retries.
   *
```

```
   * @param retries -no of retries.
   */
  public RetryableWrapper<T> withRetries(final long retries) {
    this.retries = retries;
    return this;
  }

  /**
   * Max wait time before making the next call.
   *
   * @param time - max polling interval.
   */
  public RetryableWrapper<T> maxWaitTimeInMilliSec(final long time) {
    this.maxWaitTimeInSec = time;
    return this;
  }

  /**
   * ExponentialBackoff coefficient.
   */
  public RetryableWrapper<T> withExponentialBackoff(final boolean expo) {
    this.exponentialBackoff = expo;
    return this;
  }

  /**
   * Call client method which is wrapped in submit method.
   */
  public T call() throws Exception {
    int count = 0;
    Exception exceptionForMitigationPurpose = null;
    do {
      final long waitTime = exponentialBackoff ? Math.min(getWaitTimeExp(retries),
maxWaitTimeInSec) : 0;
      try {
          return submit();
      } catch (Exception e) {
        exceptionForMitigationPurpose = e;
        if (isRetryable(e) && count < retries) {
          Thread.sleep(waitTime);
          log.debug("Retrying on exception attempt - {} on exception cause - {}",
count, e.getMessage());
        } else if (!isRetryable(e)) {
          log.error(e.getMessage());
```

```
          throw new RuntimeException(e);
        }
      }
    } while (++count < retries);

    throw new IOException(String.format(
          "Retry was unsuccessful.... attempts %d. Hence throwing exception " + "back
 to the caller...", count),
          exceptionForMitigationPurpose);
  }

  /*
   * Returns the next wait interval, in milliseconds, using an exponential backoff
   * algorithm.
   */
  private long getWaitTimeExp(final long retryCount) {
    if (0 == retryCount) {
      return 0;
    }
    return ((long) Math.pow(2, retryCount) * 100L);
  }
}
```

Définissez plusieurs propriétés à la fois à l'aide d'une seule clause SET

Au lieu d'utiliser plusieurs clauses SET pour définir des propriétés individuelles, utilisez une carte pour définir simultanément plusieurs propriétés pour une entité.

Vous pouvez utiliser :

```
MATCH (n:SomeLabel {`~id`: 'id1'})
SET n += {property1 : 'value1',
property2 : 'value2',
property3 : 'value3'}
```

Au lieu de :

```
MATCH (n:SomeLabel {`~id`: 'id1'})
SET n.property1 = 'value1'
SET n.property2 = 'value2'
SET n.property3 = 'value3'
```

La clause SET accepte une propriété unique ou une carte. Si vous mettez à jour plusieurs propriétés sur une seule entité, l'utilisation d'une seule clause SET avec une carte permet d'effectuer les mises à jour en une seule opération au lieu de plusieurs opérations, qui peuvent être exécutées plus efficacement.

Utilisez la clause SET pour supprimer plusieurs propriétés à la fois

Lorsque vous utilisez le langage OpenCypher, REMOVE est utilisé pour supprimer les propriétés d'une entité. Dans Neptune, chaque propriété supprimée nécessite une opération distincte, ce qui ajoute de la latence aux requêtes. Vous pouvez plutôt utiliser SET avec une carte pour définir les valeurs de toutes les propriétés`null`, ce qui, dans Neptune, revient à supprimer des propriétés. Neptune bénéficiera de performances accrues lorsque plusieurs propriétés d'une même entité doivent être supprimées.

Utilisez :

```
WITH {prop1: null, prop2: null, prop3: null} as propertiesToRemove
MATCH (n)
SET n += propertiesToRemove
```

Au lieu de :

```
MATCH (n)
REMOVE n.prop1, n.prop2, n.prop3
```

Utilisation des requêtes paramétrées

Il est recommandé de toujours utiliser des requêtes paramétrées lorsque vous utilisez OpenCypher. Le moteur de requêtes peut exploiter des requêtes paramétrées répétées pour des fonctionnalités telles que le cache de plans de requêtes, où l'invocation répétée de la même structure paramétrée avec des paramètres différents peut tirer parti des plans mis en cache. Le plan de requête généré pour les requêtes paramétrées est mis en cache et réutilisé uniquement lorsqu'il est terminé dans les 100 ms et que les types de paramètres sont NUMBER, BOOLEAN ou STRING.

Utilisez :

```
MATCH (n:foo) WHERE id(n) = $id RETURN n
```

Avec paramètres :

```
parameters={"id": "first"}
parameters={"id": "second"}
parameters={"id": "third"}
```

Au lieu de :

```
MATCH (n:foo) WHERE id(n) = "first" RETURN n
MATCH (n:foo) WHERE id(n) = "second" RETURN n
MATCH (n:foo) WHERE id(n) = "third" RETURN n
```

Utilisez des cartes aplaties au lieu de cartes imbriquées dans la clause UNWIND

Une structure profondément imbriquée peut limiter la capacité du moteur de requêtes à générer un plan de requête optimal. Pour pallier partiellement ce problème, les modèles définis ci-dessous créeront des plans optimaux pour les scénarios suivants :

- Scénario 1 : DÉTENDEZ-VOUS avec une liste de littéraux chiffrés, qui inclut NUMBER, STRING et BOOLEAN.
- Scénario 2 : DÉTENDEZ-VOUS avec une liste de cartes aplaties, qui inclut uniquement des littéraux chiffrés (NUMBER, STRING, BOOLEAN) comme valeurs.

Lorsque vous rédigez une requête contenant la clause UNWIND, suivez les recommandations ci-dessus pour améliorer les performances.

Exemple de scénario 1 :

```
UNWIND $ids as x
MATCH(t:ticket {`~id`: x})
```

Avec paramètres :

```
parameters={
  "ids": [1, 2, 3]
}
```

Un exemple pour le scénario 2 consiste à générer une liste de nœuds à CRÉER ou à FUSIONNER. Au lieu d'émettre plusieurs instructions, utilisez le modèle suivant pour définir les propriétés sous la forme d'un ensemble de cartes aplaties :

```
UNWIND $props as p
CREATE(t:ticket {title: p.title, severity:p.severity})
```

Avec paramètres :

```
parameters={
  "props": [
    {"title": "food poisoning", "severity": "2"},
    {"title": "Simone is in office", "severity": "3"}
  ]
}
```

Au lieu d'objets de nœuds imbriqués tels que :

```
UNWIND $nodes as n
CREATE(t:ticket n.properties)
```

Avec paramètres :

```
parameters={
  "nodes": [
    {"id": "ticket1", "properties": {"title": "food poisoning", "severity": "2"}},
    {"id": "ticket2", "properties": {"title": "Simone is in office", "severity": "3"}}
  ]
}
```

Placez des nœuds plus restrictifs sur le côté gauche dans les expressions VLP (Variable-Length Path)

Dans les requêtes VLP (Variable-Length Path), le moteur de requêtes optimise l'évaluation en choisissant de démarrer la traversée à gauche ou à droite de l'expression. La décision est basée sur la cardinalité des motifs à gauche et à droite. La cardinalité est le nombre de nœuds correspondant au modèle spécifié.

- Si le motif droit a une cardinalité de un, le côté droit sera le point de départ.

- Si le côté gauche et le côté droit ont une cardinalité égale à un, l'expansion est vérifiée des deux côtés et commence du côté dont l'expansion est la plus faible. L'extension est le nombre d'arêtes sortantes ou entrantes pour le nœud de gauche et le nœud de droite de l'expression VLP. Cette partie de l'optimisation n'est utilisée que si la relation VLP est unidirectionnelle et si le type de relation est fourni.
- Sinon, le côté gauche sera le point de départ.

Pour une chaîne d'expressions VLP, cette optimisation ne peut être appliquée qu'à la première expression. Les autres VLPs sont évalués en commençant par le côté gauche. Par exemple, supposons que la cardinalité de (a), (b) soit un et que la cardinalité de (c) soit supérieure à un.

- `(a)-[*1..]->(c)`: L'évaluation commence par (a).
- `(c)-[*1..]->(a)`: L'évaluation commence par (a).
- `(a)-[*1..]-(c)`: L'évaluation commence par (a).
- `(c)-[*1..]-(a)`: L'évaluation commence par (a).

Supposons maintenant que les arêtes entrantes de (a) soient deux, les arêtes sortantes de (a) trois, les arêtes entrantes de (b) quatre et les arêtes sortantes de (b) cinq.

- `(a)-[*1..]->(b)`: L'évaluation commence par (a) car les arêtes sortantes de (a) sont inférieures aux arêtes entrantes de (b).
- `(a)<-[*1..]-(b)`: L'évaluation commence par (a) car les arêtes entrantes de (a) sont inférieures aux arêtes sortantes de (b).

En règle générale, placez le modèle le plus restrictif sur le côté gauche d'une expression VLP.

Évitez les vérifications redondantes des étiquettes des nœuds en utilisant des noms de relations granulaires

Lors de l'optimisation des performances, l'utilisation d'étiquettes de relation exclusives aux modèles de nœuds permet de supprimer le filtrage des étiquettes sur les nœuds. Prenons l'exemple d'un modèle de graphe dans lequel la relation `likes` est uniquement utilisée pour définir une relation entre deux `person` nœuds. Nous pourrions écrire la requête suivante pour trouver ce modèle :

```
MATCH (n:person)-[:likes]->(m:person)
```

```
RETURN n, m
```

La vérification des `person` étiquettes sur n et m est redondante, car nous avons défini la relation de manière à ce qu'elle n'apparaisse que lorsque les deux sont du même type`person`. Pour optimiser les performances, nous pouvons écrire la requête comme suit :

```
MATCH (n)-[:likes]->(m)
RETURN n, m
```

Ce modèle peut également s'appliquer lorsque les propriétés sont exclusives à une étiquette de nœud unique. Supposons que seuls `person` les nœuds possèdent cette propriété`email`. Il est donc redondant de vérifier que l'étiquette du nœud `person` correspond. Écrire cette requête sous la forme :

```
MATCH (n:person)
WHERE n.email = 'xxx@gmail.com'
RETURN n
```

Est moins efficace que d'écrire cette requête sous la forme :

```
MATCH (n)
WHERE n.email = 'xxx@gmail.com'
RETURN n
```

Vous ne devez adopter ce modèle que lorsque les performances sont importantes et que votre processus de modélisation est contrôlé pour vous assurer que ces étiquettes de bord ne sont pas réutilisées pour des modèles impliquant d'autres étiquettes de nœuds. Si vous introduisez ultérieurement une `email` propriété sur une autre étiquette de nœud`company`, par exemple, les résultats seront différents entre ces deux versions de la requête.

Spécifiez les étiquettes de bord dans la mesure du possible

Il est recommandé de fournir une étiquette de bord lorsque cela est possible lorsque vous spécifiez une arête dans un motif. Prenons l'exemple de requête suivant, qui est utilisé pour relier toutes les personnes vivant dans une ville à toutes les personnes qui ont visité cette ville.

```
MATCH (person)-->(city {country: "US"})-->(anotherPerson)
RETURN person, anotherPerson
```

Si votre modèle de graphe relie des personnes à des nœuds autres que des villes à l'aide de plusieurs étiquettes de bord, Neptune devra évaluer d'autres chemins qui seront ensuite supprimés en omettant de spécifier l'étiquette finale. Dans la requête ci-dessus, aucune étiquette de bord n'ayant été donnée, le moteur effectue d'abord un travail supplémentaire, puis filtre les valeurs pour obtenir le résultat correct. Une meilleure version de la requête ci-dessus pourrait être :

```
MATCH (person)-[:livesIn]->(city {country: "US"})-[:visitedBy]->(anotherPerson)
RETURN person, anotherPerson
```

Cela facilite non seulement l'évaluation, mais permet également au planificateur de requêtes de créer de meilleurs plans. Vous pouvez même associer cette bonne pratique à des vérifications redondantes de l'étiquette des nœuds pour supprimer la vérification de l'étiquette de la ville et écrire la requête sous la forme suivante :

```
MATCH (person)-[:livesIn]->({country: "US"})-[:visitedBy]->(anotherPerson)
RETURN person, anotherPerson
```

Évitez d'utiliser la clause WITH dans la mesure du possible

La clause WITH d'OpenCypher agit comme une limite où tout ce qui se trouve avant son exécution, puis les valeurs qui en résultent, sont transmis aux parties restantes de la requête. La clause WITH est nécessaire lorsque vous avez besoin d'une agrégation temporaire ou que vous souhaitez limiter le nombre de résultats, mais en dehors de cela, vous devez essayer d'éviter d'utiliser la clause WITH. Le conseil général est de supprimer ces simples clauses WITH (sans agrégation, ordre par ou limite) pour permettre au planificateur de requêtes de travailler sur l'ensemble de la requête afin de créer un plan global optimal. Par exemple, supposons que vous ayez écrit une requête pour renvoyer toutes les personnes vivant dans `India` :

```
MATCH (person)-[:lives_in]->(city)
WITH person, city
MATCH (city)-[:part_of]->(country {name: 'India'})
RETURN collect(person) AS result
```

Dans la version ci-dessus, la clause WITH restreint le placement du motif `(city)-[:part_of]->(country {name: 'India'})` (ce qui est plus restrictif) avant`(person)-[:lives_in]->(city)`. Cela rend le plan sous-optimal. Une optimisation de cette requête serait de supprimer la clause WITH et de laisser le planificateur calculer le meilleur plan.

```
MATCH (person)-[:lives_in]->(city)
MATCH (city)-[:part_of]->(country {name: 'India'})
RETURN collect(person) AS result
```

Placez les filtres restrictifs le plus tôt possible dans la requête

Dans tous les scénarios, le placement précoce de filtres dans la requête permet de réduire les solutions intermédiaires qu'un plan de requête doit prendre en compte. Cela signifie que moins de mémoire et de ressources de calcul sont nécessaires pour exécuter la requête.

L'exemple suivant vous aide à comprendre ces impacts. Supposons que vous écriviez une requête pour renvoyer toutes les personnes qui y vivent`India`. L'une des versions de la requête peut être :

```
MATCH (n)-[:lives_in]->(city)-[:part_of]->(country)
WITH country, collect(n.firstName + " "  + n.lastName) AS result
WHERE country.name = 'India'
RETURN result
```

La version ci-dessus de la requête n'est pas le moyen le plus optimal pour réaliser ce cas d'utilisation. Le filtre `country.name = 'India'` apparaît ultérieurement dans le modèle de requête. Il collectera d'abord toutes les personnes et leur lieu de résidence, puis les regroupera par pays, puis filtrera uniquement pour le groupe pour`country.name = India`. La méthode optimale pour interroger uniquement les personnes vivant sur place, `India` puis effectuer l'agrégation des collectes.

```
MATCH (n)-[:lives_in]->(city)-[:part_of]->(country)
WHERE country.name = 'India'
RETURN collect(n.firstName + " "  + n.lastName) AS result
```

Une règle générale consiste à placer un filtre dès que possible après l'introduction de la variable.

Vérifiez explicitement si les propriétés existent

Selon la sémantique d'OpenCypher, l'accès à une propriété équivaut à une jointure facultative et doit conserver toutes les lignes même si la propriété n'existe pas. Si vous savez, sur la base de votre schéma graphique, qu'une propriété particulière existera toujours pour cette entité, le fait de vérifier explicitement l'existence de cette propriété permet au moteur de requêtes de créer des plans optimaux et d'améliorer les performances.

Prenons l'exemple d'un modèle de graphe dans lequel les nœuds de type ont `person` toujours une propriété`name`. Au lieu de faire ceci :

```
MATCH (n:person)
RETURN n.name
```

Vérifiez explicitement l'existence de la propriété dans la requête avec une vérification IS NOT NULL :

```
MATCH (n:person)
WHERE n.name IS NOT NULL
RETURN n.name
```

N'utilisez pas de chemin nommé (sauf si cela est obligatoire)

Le chemin indiqué dans une requête entraîne toujours un coût supplémentaire, ce qui peut entraîner des pénalités en termes de latence et d'utilisation de la mémoire plus élevées. Considérons la requête suivante :

```
MATCH p = (n)-[:commentedOn]->(m)
WITH p, m, n, n.score + m.score as total
WHERE total > 100
MATCH (m)-[:commentedON]->(o)
WITH p, m, n, distinct(o) as o1
RETURN p, m.name, n.name, o1.name
```

Dans la requête ci-dessus, en supposant que nous voulions uniquement connaître les propriétés des nœuds, l'utilisation du chemin « p » n'est pas nécessaire. En spécifiant le chemin nommé sous forme de variable, l'opération d'agrégation utilisant DISTINCT deviendra coûteuse à la fois en termes de temps et d'utilisation de la mémoire. Une version plus optimisée de la requête ci-dessus pourrait être :

```
MATCH (n)-[:commentedOn]->(m)
WITH m, n, n.score + m.score as total
WHERE total > 100
MATCH (m)-[:commentedON]->(o)
WITH m, n, distinct(o) as o1
RETURN m.name, n.name, o1.name
```

Évitez COLLECT (DISTINCT ())

> **Note**
>
> À partir de la version 1.4.7.0 du moteur, cette réécriture recommandée n'est plus nécessaire.

COLLECT (DISTINCT ()) est utilisé chaque fois qu'une liste contenant des valeurs distinctes doit être formée. COLLECT est une fonction d'agrégation, et le regroupement est effectué sur la base de clés supplémentaires projetées dans la même instruction. Lorsque distinct est utilisé, l'entrée est divisée en plusieurs segments, chaque segment désignant un groupe à réduire. Les performances seront affectées à mesure que le nombre de groupes augmentera. Dans Neptune, il est beaucoup plus efficace d'exécuter DISTINCT avant collecting/forming la liste. Cela permet d'effectuer le regroupement directement sur les touches de regroupement pour l'ensemble du morceau.

Considérons la requête suivante :

```
MATCH (n:Person)-[:commented_on]->(p:Post)
WITH n, collect(distinct(p.post_id)) as post_list
RETURN n, post_list
```

Une manière plus optimale d'écrire cette requête est la suivante :

```
MATCH (n:Person)-[:commented_on]->(p:Post)
WITH DISTINCT n, p.post_id as postId
WITH n, collect(postId) as post_list
RETURN n, post_list
```

Préférez la fonction de propriétés à la recherche de propriétés individuelle lors de la récupération de toutes les valeurs de propriété

La `properties()` fonction est utilisée pour renvoyer une carte contenant toutes les propriétés d'une entité, et elle est bien plus efficace que de renvoyer des propriétés individuellement.

En supposant que vos `Person` nœuds contiennent 5 propriétés `firstName` `lastNameage`,,`dept`,`company`, et que la requête suivante serait préférable :

```
MATCH (n:Person)
WHERE n.dept = 'AWS'
```

```
RETURN properties(n) as personDetails
```

Plutôt que d'utiliser :

```
MATCH (n:Person)
WHERE n.dept = 'AWS'
RETURN n.firstName, n.lastName, n.age, n.dept, n.company

=== OR ===

MATCH (n:Person)
WHERE n.dept = 'AWS'
RETURN {firstName: n.firstName, lastName: n.lastName, age: n.age,
department: n.dept, company: n.company} as personDetails
```

Effectuer des calculs statiques en dehors de la requête

Il est recommandé de résoudre les calculs statiques (mathematical/string opérations simples) côté client. Prenons cet exemple où vous souhaitez rechercher toutes les personnes âgées d'un an ou moins que l'auteur :

```
MATCH (m:Message)-[:HAS_CREATOR]->(p:person)
WHERE p.age <= ($age + 1)
RETURN m
```

Ici, `$age` est injecté dans la requête via des paramètres, puis ajouté à une valeur fixe. Cette valeur est ensuite comparée à`p.age`. Au lieu de cela, une meilleure approche consisterait à effectuer l'ajout côté client et à transmettre la valeur calculée sous forme de paramètre $ageplusone. Cela permet au moteur de requêtes de créer des plans optimisés et d'éviter les calculs statiques pour chaque ligne entrante. En suivant ces directives, une version plus efficace de la requête serait :

```
MATCH (m:Message)-[:HAS_CREATOR]->(p:person)
WHERE p.age <= $ageplusone
RETURN m
```

Entrées par lots utilisant UNWIND au lieu d'instructions individuelles

Chaque fois que la même requête doit être exécutée pour différentes entrées, au lieu d'exécuter une requête par entrée, il serait beaucoup plus performant d'exécuter une requête pour un lot d'entrées.

Si vous souhaitez effectuer une fusion sur un ensemble de nœuds, l'une des options consiste à exécuter une requête de fusion par entrée :

```
MERGE (n:Person {`~id`: $id})
SET n.name = $name, n.age = $age, n.employer = $employer
```

Avec paramètres :

```
params = {id: '1', name: 'john', age: 25, employer: 'Amazon'}
```

La requête ci-dessus doit être exécutée pour chaque entrée. Bien que cette approche fonctionne, elle peut nécessiter l'exécution de nombreuses requêtes pour un grand nombre d'entrées. Dans ce scénario, le traitement par lots peut contribuer à réduire le nombre de requêtes exécutées sur le serveur et à améliorer le débit global.

Utilisez le modèle suivant :

```
UNWIND $persons as person
MERGE (n:Person {`~id`: person.id})
SET n += person
```

Avec paramètres :

```
params = {persons: [{id: '1', name: 'john', age: 25, employer: 'Amazon'},
{id: '2', name: 'jack', age: 28, employer: 'Amazon'},
{id: '3', name: 'alice', age: 24, employer: 'Amazon'}...]}
```

Il est recommandé d'expérimenter avec différentes tailles de lots afin de déterminer ce qui convient le mieux à votre charge de travail.

Préférez utiliser la personnalisation IDs pour le nœud ou la relation

Neptune permet aux utilisateurs d'attribuer des nœuds et des relations IDs de manière explicite. L'identifiant doit être globalement unique dans l'ensemble de données et déterministe pour être utile. Un identifiant déterministe peut être utilisé comme mécanisme de recherche ou de filtrage, tout comme les propriétés ; toutefois, l'utilisation d'un identifiant est beaucoup plus optimisée du point de vue de l'exécution des requêtes que celle des propriétés. L'utilisation de la personnalisation présente plusieurs avantages IDs :

- Les propriétés peuvent être nulles pour une entité existante, mais l'ID doit exister. Cela permet au moteur de requête d'utiliser une jointure optimisée lors de l'exécution.
- Lorsque des requêtes de mutation simultanées sont exécutées, les risques d'exceptions de modification simultanées (CMEs) sont considérablement réduits lorsqu'elles IDs sont utilisées pour accéder à des nœuds, car le nombre de verrous requis est inférieur à IDs celui des propriétés en raison de leur caractère unique imposé.
- L'utilisation IDs permet d'éviter de créer des doublons de données, car Neptune impose l'unicité, contrairement aux IDs propriétés.

L'exemple de requête suivant utilise un identifiant personnalisé :

> **Note**
>
> La propriété `~id` est utilisée pour spécifier l'ID, alors qu'elle `id` est simplement stockée comme n'importe quelle autre propriété.

```
CREATE (n:Person {`~id`: '1', name: 'alice'})
```

Sans utiliser d'identifiant personnalisé :

```
CREATE (n:Person {id: '1', name: 'alice'})
```

Si vous utilisez ce dernier mécanisme, il n'y a pas d'application de l'unicité et vous pouvez exécuter la requête ultérieurement :

```
CREATE (n:Person {id: '1', name: 'john'})
```

Cela crée un deuxième nœud `id=1` nommé`john`. Dans ce scénario, vous auriez maintenant deux nœuds portant `id=1` chacun un nom différent - (alice et john).

Évitez de faire des calculs ~id dans la requête

Lorsque vous utilisez la personnalisation IDs dans les requêtes, effectuez toujours des calculs statiques en dehors des requêtes et fournissez ces valeurs dans les paramètres. Lorsque des valeurs statiques sont fournies, le moteur est mieux à même d'optimiser les recherches et d'éviter de scanner et de filtrer ces valeurs.

Si vous souhaitez créer des limites entre les nœuds existants dans la base de données, l'une des options suivantes peut être la suivante :

```
UNWIND $sections as section
MATCH (s:Section {`~id`: 'Sec-' + section.id})
MERGE (s)-[:IS_PART_OF]->(g:Group {`~id`: 'g1'})
```

Avec paramètres :

```
parameters={sections: [{id: '1'}, {id: '2'}]}
```

Dans la requête ci-dessus, `id` la section est calculée dans la requête. Comme le calcul est dynamique, le moteur ne peut pas insérer les identifiants de manière statique et finit par scanner tous les nœuds de section. Le moteur effectue ensuite un post-filtrage pour les nœuds requis. Cela peut être coûteux si la base de données contient de nombreux nœuds de section.

Une meilleure façon d'y parvenir est d'avoir introduit au `Sec-` préalable les identifiants transmis à la base de données :

```
UNWIND $sections as section
MATCH (s:Section {`~id`: section.id})
MERGE (s)-[:IS_PART_OF]->(g:Group {`~id`: 'g1'})
```

Avec paramètres :

```
parameters={sections: [{id: 'Sec-1'}, {id: 'Sec-2'}]}
```

Mise à jour/fusion de plusieurs nœuds

Lorsque vous exécutez `MERGE` ou `CREATE` interrogez plusieurs nœuds, il est recommandé d'utiliser une `UNWIND` clause combinée avec une seule MERGE/CREATE clause plutôt qu'une MERGE/CREATE clause pour chaque nœud. Les requêtes qui utilisent une clause pour un nœud entraînent un plan d'exécution inefficace car chaque ligne doit être optimisée. Cela signifie que la majeure partie du temps d'exécution de la requête est consacrée au traitement statique plutôt qu'à la mise à jour proprement dite.

Une clause par nœud n'est pas optimale car elle ne s'adapte pas à l'augmentation du nombre de nœuds :

```
MERGE (p1:Person {name: 'NameA'})
```

```
ON CREATE SET p1 += {prop1: 'prop1V1', prop2: 'prop2V1'}
MERGE (p2:Person {name: 'NameB'})
ON CREATE SET p2 += {prop1: 'prop1V2', prop2: 'prop2V2'}
MERGE (p3:Person {name: 'NameC'})
ON CREATE SET p3 += {prop1: 'prop1V3', prop2: 'prop1V3'}
```

L'utilisation d'une clause UNWIND en conjonction avec une MERGE/CREATE clause permet le même comportement mais un plan d'exécution plus optimal. Dans cet esprit, la requête modifiée ressemblerait à ce qui suit :

```
## If not using custom id for nodes/relationship
UNWIND [{name: 'NameA', prop1: 'prop1V1', prop2: 'prop2V1'}, {name: 'NameB', prop1:
 'prop1V2', prop2: 'prop2V2'}, {name: 'NameC', prop1: 'prop1V3', prop2: 'prop1V3'}] AS
 props
MERGE (p:Person {name: props.name})
ON CREATE SET p = props

## If using custom id for nodes/relationship
UNWIND [{`~id`: '1', 'name': 'NameA', 'prop1: 'prop1V1', prop2: 'prop2V1'}, {`~id`:
 '2', name: 'NameB', prop1: 'prop1V2', prop2: 'prop2V2'}, {`~id`: '3', name: 'NameC',
 prop1: 'prop1V3', prop2: 'prop1V3'}] AS props
MERGE (p:Person {`~id`: props.id})
ON CREATE SET p = removeKeyFromMap(props, '~id')
```

Bonnes pratiques Neptune avec l'utilisation de SPARQL

Suivez ces bonnes pratiques lorsque vous utilisez le langage de requête SPARQL avec Neptune. Pour plus d'informations sur l'utilisation de SPARQL dans Neptune, consultez Accès au graphe Neptune avec SPARQL.

Rubriques

- Interrogation de tous les graphes par défaut
- Spécification d'un graphe nommé pour le chargement
- Choisir entre FILTER, FILTER...IN et VALUES dans vos requêtes

Interrogation de tous les graphes par défaut

Amazon Neptune associe chaque triplet à un graphe nommé. Le graphe par défaut est défini comme l'union de tous les graphes nommés.

Si vous envoyez une requête SPARQL sans spécifier explicitement un graphe via le mot-clé `GRAPH` ou des constructions telles que `FROM NAMED`, Neptune prend toujours en compte tous les triplets dans votre instance de base de données. Par exemple, la requête suivante renvoie tous les triplets à partir d'un point de terminaison Neptune SPARQL :

```
SELECT * WHERE { ?s ?p ?o }
```

Les triplets qui apparaissent dans plusieurs graphes ne sont renvoyés qu'une seule fois.

Pour plus d'informations sur la spécification du graphe par défaut, consultez la section RDF Dataset dans la spécification de langage de requête SPARQL 1.1.

Spécification d'un graphe nommé pour le chargement

Amazon Neptune associe chaque triplet à un graphe nommé. Si vous ne spécifiez pas un graphe nommé lors du chargement, de l'insertion ou de la mise à jour de triplets, Neptune utilise le graphe nommé de secours défini par l'URI, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Si vous utilisez le chargeur en bloc Neptune, vous pouvez spécifier le graphe nommé pour utiliser tous les triplets (ou quadrilatères avec la quatrième position vide) à l'aide du paramètre `parserConfiguration: namedGraphUri`. Pour plus d'informations sur la syntaxe de la commande `Load` du chargeur Neptune, consultez the section called “Commande Loader”.

Choisir entre FILTER, FILTER...IN et VALUES dans vos requêtes

Il existe trois façons de base d'injecter des valeurs dans les requêtes SPARQL : `FILTER`, `FILTER...IN` et `VALUES`.

Par exemple, imaginons que vous souhaitiez rechercher les amis de plusieurs personnes dans une seule requête. En utilisant `FILTER`, vous pouvez structurer votre requête comme suit :

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
WHERE {?s foaf:knows ?o. FILTER (?s = ex:person1 || ?s = ex:person2)}
```

Cette opération renvoie tous les triplets du graphe dans lesquels `?s` est lié à `ex:person1` ou `ex:person2` et qui ont un arc sortant intitulé `foaf:knows`.

Vous pouvez également créer une requête en utilisant `FILTER...IN` qui renvoie des résultats équivalents :

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
WHERE {?s foaf:knows ?o. FILTER (?s IN (ex:person1, ex:person2))}
```

Vous pouvez également créer une requête en utilisant `VALUES` qui, dans ce cas, renvoie également des résultats équivalents :

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
WHERE {?s foaf:knows ?o. VALUES ?s {ex:person1 ex:person2}}
```

Bien que, dans bien des cas, ces requêtes soient équivalentes sémantiquement, il y a certains cas où les deux variantes `FILTER` diffèrent de la variante `VALUES` :

- Le premier cas est lorsque vous injectez des valeurs en double, telles que l'injection de la même personne deux fois. Dans ce cas, la requête `VALUES` inclut les doublons dans votre résultat. Vous pouvez explicitement éliminer ces doublons en ajoutant un `DISTINCT` à la clause `SELECT`. Toutefois, dans certaines situations, vous souhaiterez peut-être conserver les doublons dans les résultats des requêtes à des fins d'injection des valeurs redondantes.

  Cependant, les versions `FILTER...IN` et `FILTER` extraient la valeur une seule fois lorsque la même valeur apparaît plusieurs fois.
- Le deuxième cas est lié au fait que `VALUES` génère toujours une correspondance exacte, alors que `FILTER` pourrait appliquer une promotion de type et réaliser des correspondances partielles dans certains cas.

  Par exemple, lorsque vous incluez une valeur littérale comme `"2.0"^^xsd:float` dans votre clause VALUES, une requête `VALUES` établit une correspondance exacte avec cette valeur littérale, y compris la valeur littérale et le type de données.

En revanche, `FILTER` produit une correspondance partielle pour ces littéraux numériques. Les correspondances peuvent inclure des littéraux avec la même valeur mais des types de données numériques, par exemple `xsd:double`.

> **Note**
>
> Il n'y a aucune différence entre le `VALUES` comportement `FILTER` et lors de l'énumération des littéraux de chaîne ou. URIs

Les différences entre `FILTER` et `VALUES` peuvent affecter l'optimisation et la stratégie d'évaluation de requête qui en résulte. Sauf si votre cas d'utilisation nécessite des correspondances partielles, nous vous recommandons d'utiliser `VALUES` car cela évite d'examiner les cas spéciaux relatifs au transtypage. En conséquence, `VALUES` produit souvent une requête plus efficace qui s'exécute plus rapidement et qui coûte moins cher.

Bonnes pratiques d'utilisation de Neptune avec Amazon ECS et Amazon EKS

L'exécution de votre application graphique dans des conteneurs sur Amazon Elastic Container Service (Amazon ECS) ou Amazon Elastic Kubernetes Service (Amazon EKS) introduit des considérations qui ne s'appliquent pas aux déploiements de serveurs traditionnels de longue durée. Les conteneurs sont éphémères : ils démarrent, s'arrêtent et se remplacent lors de mises à jour continues et d'événements de dimensionnement. Ce cycle de vie affecte la façon dont votre application gère les connexions à Neptune, gère les remplacements d'hôtes et s'authentifie auprès d'IAM.

Les sections suivantes décrivent les considérations pratiques relatives à l'exécution des charges de travail Neptune dans des environnements conteneurisés.

Rubriques

- Gérez les pools de connexions et le cycle de vie dans les environnements conteneurisés
- Gérer le remplacement de l'hôte et le blocage de la connexion
- Configuration du réseau, des groupes de sécurité et de l'authentification IAM

Gérez les pools de connexions et le cycle de vie dans les environnements conteneurisés

La gestion efficace des connexions à Neptune est essentielle dans les environnements conteneurisés où les conteneurs (pods dans Kubernetes) démarrent, s'arrêtent et évoluent fréquemment. Neptune impose plusieurs WebSocket limites de connexion :

- Chaque type d'instance dispose d'un nombre maximum de WebSocket connexions simultanées
- Les connexions inactives sont fermées après 20 à 25 minutes d'inactivité
- Avec l'authentification IAM, les connexions sont fermées de force après 10 jours, quelle que soit l'activité (voir WebSocket les limites de connexion)

Certains pilotes prennent en charge les pulsations cardiaques continues afin d'éviter les déconnexions inactives. Par exemple, le pilote Java Gkremlin `keepAliveInterval` envoie un ping périodique au serveur. Consultez les options de configuration spécifiques à votre pilote pour déterminer quels paramètres de maintien en vie sont disponibles et comment ils interagissent avec le délai d'inactivité de Neptune.

La compréhension de ces limites vous permet de concevoir une gestion des connexions qui fonctionne bien avec les modèles d'orchestration des conteneurs tels que les déploiements progressifs, le dimensionnement automatique et le recyclage des conteneurs. Pour obtenir des instructions générales sur la gestion des connexions avec le pilote Java Gkremlin, reportez-vous aux sections Fermeture du client pour éviter de limiter les connexions etCréation d'une connexion après un basculement.

Fermer les connexions lors de l'arrêt du conteneur

Implémentez un arrêt progressif en ajoutant un `SIGTERM` gestionnaire qui ferme votre client graphique avant la sortie du conteneur. Par exemple, appelez Gemp's Java`cluster.close()`, Garmlin's `driverRemoteConnection.close()` in Go ou Python, ou Bolt `driver.close()` pour le pilote Neo4j. Dans Kubernetes, utilisez un `preStop` crochet pour vider les connexions avant `SIGTERM` leur envoi, en particulier lors des mises à jour continues.

> **Note**
>
> Docker et Amazon ECS livrent uniquement `SIGTERM` au PID 1 dans le conteneur. Utilisez le formulaire d'exécution dans votre Dockerfile (par exemple, `ENTRYPOINT ["java",`

`"-jar", "app.jar"]` non`ENTRYPOINT java -jar app.jar`) ou un processus d'initialisation pour vous `tini` assurer que votre application reçoit le signal.

Utiliser une seule instance client par conteneur

La création de plusieurs `neo4j.Driver` instances de `DriverRemoteConnection` Gkremlin ou de Bolt dans le même conteneur multiplie l'utilisation des connexions. Partagez une instance client unique sur tous les threads du conteneur. Les objets clients sont sûrs dans un contexte multithread.

Dimensionnez les pools de connexions par rapport à votre flotte

Configurez de `maxConnectionPoolSize` telle sorte que le nombre total de connexions entre tous les conteneurs ne dépasse pas la limite de connexion à l'instance Neptune. Par exemple, si vous exécutez 20 conteneurs avec 8 threads chacun, calculez la taille du pool par conteneur de manière à ce que 20 × la taille du pool reste dans la limite d'instance pour votre type d'instance. Pour plus d'informations sur la configuration de la taille des pools de connexions, consultezDéfinition de `maxInProcessPerConnection` et `maxSimultaneousUsagePerConnection` sur la même valeur.

Définir un délai d'attente pour la connexion

Dans le pilote Java Gkremlin, configurez `maxWaitForConnection` à une valeur raisonnable, telle que 5 à 10 secondes. La valeur par défaut est souvent infinie ou très longue, ce qui entraîne le blocage des threads lorsque le pool est épuisé au lieu d'échouer rapidement avec une erreur claire. Tous les pilotes de langue ne prennent pas en charge ce paramètre. Consultez la documentation de votre pilote pour connaître l'option de configuration équivalente.

Gérer le remplacement de l'hôte et le blocage de la connexion

Lorsque Neptune remplace un hôte (par exemple, lors d'une maintenance ou d'un basculement), les connexions existantes à cet hôte ne sont plus valides. Dans les environnements conteneurisés, cela peut bloquer tous les threads d'un conteneur si le client ne gère pas le remplacement correctement.

Utiliser les versions actuelles du client

Si vous utilisez le langage de requête Gremlin, utilisez une version TinkerPop du pilote compatible avec la version de votre moteur Neptune (Accès au graphe Neptune avec Gremlinvoir le tableau de compatibilité). Si vous utilisez le pilote Java, envisagez `neptune-gremlin-client` d'ajouter une enveloppe au pilote TinkerPop Java qui ajoute des fonctionnalités de gestion des connexions, telles

que la vérification de l'état des terminaux et la gestion du basculement. Il suit les mêmes règles de compatibilité de version que le TinkerPop pilote sous-jacent.

Utilisez `neptune-gremlin-client` la version 3.x (ou au minimum la version 2.0.7), selon ce que votre version de Neptune autorise. Ces nouvelles versions améliorent la résilience et la gestion des connexions.

Pour les utilisateurs d'OpenCypher dotés du pilote Neo4j, fermez et recréez l'`Driver`objet lorsque vous détectez un échec de connexion lors du basculement. Neptune prend en charge les versions 1 à 4.0 du protocole Bolt. Pour de plus amples informations, veuillez consulter Bonnes pratiques Neptune avec openCypher et Bolt.

Utiliser les points de terminaison du cluster ou du lecteur

Ne vous connectez pas directement aux points de terminaison de l'instance. Utilisez le point de terminaison du cluster pour les écritures et le point de terminaison du lecteur pour les lectures. Si vous devez utiliser des points de terminaison d'instance avec`neptune-gremlin-client`, activez le filtrage de vérification de l'état des points de terminaison via l'`/status`API.

Configuration des sondes de vivacité avec tolérance

Réglez votre sonde de vivacité Kubernetes sur `failureThreshold` au moins 30 avec une période de 10 secondes (300 secondes au total). Cela empêche Kubernetes de redémarrer les pods pendant la période d'environ 5 minutes pendant laquelle Neptune effectue le remplacement d'un hôte.

Implémenter une nouvelle tentative avec backoff

Une seule demande échouée lors du remplacement de l'hôte ne devrait pas faire planter le conteneur. Implémentez une logique de nouvelle tentative avec un retard exponentiel en cas de défaillance de connexion afin que les erreurs transitoires lors du remplacement soient résolues sans intervention. Pour obtenir des conseils sur les exceptions réessayables, consultez la section Exceptions relatives aux transactions Neptune.

Configuration du réseau, des groupes de sécurité et de l'authentification IAM

Les déploiements conteneurisés sur Amazon ECS ou Amazon EKS nécessitent une configuration réseau correcte et une gestion des informations d'identification IAM pour se connecter à Neptune.

Liste de contrôle des groupes de sécurité

- Le groupe de sécurité du cluster Neptune doit autoriser le TCP entrant sur le port 8182 (ou le port personnalisé configuré pour votre cluster Neptune) depuis la plage d'adresses CIDR du nœud ou du pod Amazon EKS.
- Si vous utilisez le plug-in Amazon VPC CNI avec une mise en réseau personnalisée, vérifiez que le sous-réseau pod possède une route vers le sous-réseau Neptune.
- Neptune doit se trouver dans le même Amazon VPC que votre cluster Amazon ECS ou Amazon EKS, ou être connecté via Amazon VPC peering ou Transit Gateway.

Vérifiez la connectivité depuis l'intérieur du conteneur

Avant de résoudre les problèmes au niveau de l'application, confirmez le chemin réseau en exécutant l'une des commandes suivantes depuis le conteneur :

AWS CLI

```
aws neptunedata get-engine-status \
  --endpoint-url https://your-neptune-endpoint:8182
```

Pour plus d'informations, consultez get-engine-status dans la référence des commandes. AWS CLI

SDK

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    region_name='us-east-1',
    endpoint_url='https://your-neptune-endpoint:8182',
    config=Config(read_timeout=10)
)

response = client.get_engine_status()
print(response['status'])
```

awscurl

```
awscurl https://your-neptune-endpoint:8182/status \
  --region us-east-1 \
```

```
  --service neptune-db
```

> **Note**
>
> Cet exemple suppose que vos AWS informations d'identification sont configurées dans votre environnement. Remplacez *us-east-1* par la région de votre cluster Neptune.

curl

```
curl -k https://your-neptune-endpoint:8182/status
```

> **Note**
>
> Le `-k` drapeau ignore la vérification du certificat. Utilisez-le uniquement pour les tests de connectivité, pas dans le code de production.

Une réponse réussie confirme que les groupes de sécurité, le routage et la résolution DNS fonctionnent tous. Un délai d'attente indique un problème au niveau du réseau.

Actualisation des informations d'identification IAM

Si vous utilisez l'authentification IAM avec Neptune, vos pods doivent actualiser leurs informations d'identification avant leur expiration. Nous recommandons Amazon EKS Pod Identity pour les nouveaux déploiements. Les rôles IAM pour les comptes de service (IRSA) sont également pris en charge et largement utilisés.

Quelle que soit l'approche, récupérez les informations d'identification dans la fonction du générateur d'authentification, et non une seule fois au démarrage. Les jetons IRSA et Pod Identity expirent au bout d'une heure. Si vous mettez en cache les informations d'identification lors de l'initialisation, les demandes échoueront une `AccessDeniedException: security token expired` fois le jeton expiré.

- Pour gremlin-go : appelez à `cfg.Credentials.Retrieve(ctx)` l'intérieur de la `gen` fonction passée à. `NewDynamicAuth`
- Pour Java : utilisez `NeptuneGremlinClient` avec un `HandshakeInterceptor` qui actualise la signature SigV4 à chaque nouvelle connexion.

Cross-account accès

Neptune ne prend pas en charge l'authentification IAM directe entre comptes. Si votre cluster Amazon EKS se trouve sur un AWS compte différent de celui de votre cluster Neptune, le pod doit d'abord assumer un rôle dans le compte Neptune avant de s'authentifier.

Configuration d'entrée pour Graph Explorer

Si vous exécutez Graph Explorer dans un environnement conteneurisé, assurez-vous que tous les chemins sont accessibles au proxy, et pas seulement. `/explorer` Sur Amazon EKS, configurez les règles de votre contrôleur d'entrée pour autoriser tous les chemins. Sur Amazon ECS, configurez le groupe cible et les règles d'écoute de votre Application Load Balancer pour transférer tous les chemins vers le conteneur Graph Explorer. Le proxy de Graph Explorer passe des appels à `/summary`,`/gremlin`,`/openCypher`, et`/sparql`. Path-restricted les règles de routage interrompent le proxy en silence.

Bonnes pratiques d'utilisation du AWS CLI et SDK avec Neptune

Par défaut, les AWS SDK AWS CLI et expirent au bout de 60 secondes et réessayent automatiquement les demandes ayant échoué. Pour les requêtes dont l'exécution prend plus de 60 secondes, ce comportement par défaut entraîne une double exécution des requêtes. La nouvelle tentative déclenche une nouvelle demande alors que la requête d'origine est toujours en cours d'exécution sur le serveur.

> **Important**
>
> Lorsque vous exécutez des requêtes de longue durée via l'API du plan de données Neptune (par exemple)`execute-open-cypher-query`, désactivez les tentatives automatiques. Augmentez ou supprimez également le délai de lecture. Dans le cas contraire, le client réessaie la requête alors que l'original est toujours en cours d'exécution, ce qui double la charge sur votre cluster Neptune.

Les recommandations suivantes s'appliquent à toute opération du plan de données Neptune qui peut s'exécuter plus longtemps que le délai d'attente par défaut de 60 secondes :

- Définissez le délai de lecture sur 0 (infini) ou sur une valeur supérieure à la durée de requête la plus longue attendue.

- Définissez le nombre maximum de tentatives sur 1 (une tentative, zéro tentative). Cela empêche le client de réessayer une demande que le serveur est toujours en train de traiter.

Les exemples suivants montrent comment configurer ces paramètres pour les SDK AWS CLI les plus courants.

AWS CLI

```
export AWS_MAX_ATTEMPTS=1
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-cluster-endpoint:port \
  --open-cypher-query "RETURN 1" \
  --cli-read-timeout 0
```

Le `--cli-read-timeout 0` paramètre désactive le délai de lecture, ce qui permet d' AWS CLI attendre indéfiniment une réponse. La variable d'`AWS_MAX_ATTEMPTS=1`environnement définit le nombre maximum de tentatives à 1, ce qui signifie que la demande est essayée exactement une fois sans nouvelle tentative.

Python (boto3)

```
import boto3
from botocore.config import Config

client = boto3.client(
    'neptunedata',
    endpoint_url='https://your-cluster-endpoint:port',
    region_name='us-east-1',
    config=Config(
        retries={'total_max_attempts': 1, 'mode': 'standard'},
        read_timeout=None
    )
)
```

Le réglage `read_timeout=None` désactive le délai de lecture. Le réglage `total_max_attempts` sur 1 signifie que la demande est essayée une seule fois sans nouvelle tentative.

Java V2 SDK

```
import software.amazon.awssdk.services.neptunedata.NeptunedataClient;
```

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.regions.Region;
import java.time.Duration;
import java.net.URI;

// Build the Neptune dataplane client with no retries and infinite timeout
NeptunedataClient neptunedataClient = NeptunedataClient.builder()
    .endpointOverride(URI.create("https://your-cluster-endpoint:port"))
    .region(Region.of("us-east-1"))
    .overrideConfiguration(ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ZERO)
        .retryPolicy(RetryPolicy.none())
        .build())
    .build();
```

Le réglage `apiCallTimeout` sur `Duration.ZERO` configure un délai d'expiration infini pour l'appel d'API. L'utilisation `RetryPolicy.none()` désactive toutes les tentatives automatiques.

Limites d'Amazon Neptune

Régions

Amazon Neptune est disponible dans les régions suivantes : AWS

- USA Est (Virginie du Nord) : `us-east-1`
- USA Est (Ohio) : `us-east-2`
- USA Ouest (Californie du Nord) : `us-west-1`
- USA Ouest (Oregon) : `us-west-2`
- Canada (Centre) : `ca-central-1`
- Canada-Ouest (Calgary) : `ca-west-1`
- Amérique du Sud (São Paulo) : `sa-east-1`
- Europe (Stockholm) : `eu-north-1`
- Europe (Espagne) : `eu-south-2`
- Europe (Irlande) : `eu-west-1`
- Europe (Londres) : `eu-west-2`
- Europe (Paris) : `eu-west-3`
- Europe (Francfort) : `eu-central-1`
- Moyen-Orient (Bahreïn) : `me-south-1`
- Moyen-Orient (EAU) : `me-central-1`
- Israël (Tel Aviv) : `il-central-1`
- Afrique (Le Cap) : `af-south-1`
- Asie-Pacifique (Hong Kong) : `ap-east-1`
- Asie-Pacifique (Tokyo) : `ap-northeast-1`
- Asie-Pacifique (Séoul) : `ap-northeast-2`
- Asie-Pacifique (Osaka) : `ap-northeast-3`
- Asie-Pacifique (Singapour) : `ap-southeast-1`
- Asie-Pacifique (Sydney) : `ap-southeast-2`

- Asie-Pacifique (Jakarta) : `ap-southeast-3`
- Asie-Pacifique (Melbourne) : `ap-southeast-4`
- Asie-Pacifique (Malaisie) : `ap-southeast-5`
- Asie-Pacifique (Mumbai) : `ap-south-1`
- Asie-Pacifique (Hyderabad) : `ap-south-2`
- Chine (Beijing) : `cn-north-1`
- Chine (Ningxia) : `cn-northwest-1`
- AWS GovCloud (US-West): `us-gov-west-1`
- AWS GovCloud (US-East): `us-gov-east-1`

Différences dans les régions chinoises

Comme c'est le cas pour de nombreux AWS services, Amazon Neptune fonctionne légèrement différemment en Chine (Pékin) et en Chine (Ningxia) par rapport aux autres régions. AWS

Par exemple, lorsque Neptune ML utilise Amazon API Gateway pour créer son service d'exportation, l'authentification IAM est activée par défaut. Dans les régions chinoises, le processus de modification de cette option est légèrement différent de celui des autres régions.

Ces différences et d'autres sont expliquées ici.

Différences dans la région Asie-Pacifique (Hyderabad)

Les fonctionnalités Neptune suivantes ne sont pas prises en charge dans la région Asie-Pacifique (Hyderabad) () : `ap-south-2`

- Bases de données mondiales Neptune
- Cross-region copie d'instantanés (voirCopie d'un instantané)

Taille maximale des volumes de cluster de stockage

Un volume de cluster Neptune peut atteindre une taille maximale de 128 tebioctets (TiB) dans toutes les régions prises en charge. Consultez Stockage, fiabilité et disponibilité d'Amazon Neptune.

Tailles d'instance de base de données prises en charge

Neptune prend en charge différentes classes d'instances de base de données dans différentes régions. AWS Pour déterminer les classes qui sont prises en charge dans une région donnée, consultez Tarification Amazon Neptune et sélectionnez la région qui vous intéresse.

Limites pour chacun AWS compte

Pour certaines fonctionnalités de gestion, Amazon Neptune utilise une technologie opérationnelle partagée avec Amazon Relational Database Service (Amazon RDS).

Pour chaque AWS compte, le nombre de ressources Amazon Neptune et Amazon RDS que vous pouvez créer est limité pour chaque région. Ces ressources incluent des instances et des clusters de bases de données.

Une fois qu'une limite a été atteinte pour une ressource, les appels supplémentaires pour créer cette ressource échouent avec une exception.

Pour obtenir la liste des limites partagées entre Amazon Neptune et Amazon RDS, consultez la section Limites dans Amazon RDS dans le Guide de l'utilisateur Amazon RDS.

La connexion à Neptune nécessite un VPC

Amazon Neptune est un service de cloud privé virtuel (VPC) uniquement.

En outre, les instances n'autorisent pas l'accès depuis l'extérieur du VPC.

Neptune nécessite SSL

À partir de la version `1.0.4.0` du moteur, Amazon Neptune autorise uniquement les connexions SSL (Secure Sockets Layer) via HTTPS vers n'importe quel point de terminaison d'instance ou de cluster.

Neptune nécessite la version 1.2 du protocole TLS, avec les suites de chiffrement solides suivantes :

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`

- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

Zones de disponibilité et groupes de sous-réseaux de base de données

Amazon Neptune nécessite un groupe de sous-réseaux de base de données pour chaque cluster, avec des sous-réseaux dans au moins deux zones de disponibilité (AZ) prises en charge.

Nous vous recommandons d'utiliser trois sous-réseaux ou plus dans différentes zones de disponibilité.

Charge utile maximale des demandes HTTP (150 Mo)

La taille totale des requêtes HTTP SPARQL et Gremlin doit être inférieure à 150 Mo. Si une demande dépasse cette taille, Neptune renvoie `HTTP 400: BadRequestException`.

Cette limite ne s'applique pas aux WebSockets connexions Gremlin.

Différences d'implémentation de Gremlin

L'implémentation de Gremlin par Amazon Neptune peut se distinguer par certains détails des autres implémentations de Gremlin.

Pour de plus amples informations, veuillez consulter Conformité d'Amazon Neptune avec les normes Gremlin.

Neptune ne prend pas en charge les caractères nuls dans les chaînes de données

Neptune ne prend pas en charge les caractères nuls dans les chaînes. Cela est vrai pour les données des graphes de propriétés pour Gremlin et OpenCypher, ainsi que pour les données. RDF/SPARQL

SPARQL UPDATE LOAD from URI

`SPARQL UPDATE LOAD` à partir de l'URI fonctionne uniquement avec les ressources qui se trouvent dans le même VPC.

Cela comprend les URL Amazon S3 dans la même région que le cluster avec un point de terminaison de VPC Amazon S3 créé.

L'URL Amazon S3 doit être en HTTPS, et toute authentification doit être incluse dans l'URL. Pour plus d'informations, consultez Demandes d'authentification : utilisation des paramètres de requête dans la Référence de l'API Amazon Simple Storage Service.

Pour plus d'informations sur la création d'un point de terminaison VPC, consultez Création d'un point de terminaison de VPC Amazon S3.

Si vous devez charger les données à partir d'un fichier, nous vous recommandons d'utiliser l'API de chargeur Amazon Neptune. Pour de plus amples informations, veuillez consulter Utilisation du chargeur en vrac Amazon Neptune pour ingérer des données.

> **Note**
>
> L'API de chargeur Amazon Neptune est non ACID.

Authentification IAM et contrôle d'accès

Dans les versions du moteur Neptune antérieures à la version 1.2.0.0, l'authentification IAM et le contrôle d'accès ne sont pris en charge qu'au niveau du cluster de bases de données. Depuis la version `1.2.0.0`, vous pouvez toutefois contrôler l'accès basé sur les requêtes à un niveau plus détaillé à l'aide des clés de condition dans les politiques IAM. Pour plus d'informations, consultez Utilisation d'actions de requête dans les déclarations de stratégie d'accès aux données Neptune et Authentification de votre base de données Amazon Neptune avec Gestion des identités et des accès AWS.

La console Amazon Neptune nécessite NeptuneReadOnlyAccessdes autorisations. Vous pouvez restreindre l'accès aux utilisateurs IAM en révoquant cet accès. Pour de plus amples informations, consultez Utilisation AWS politiques gérées pour accéder aux bases de données Amazon Neptune.

Amazon Neptune ne prend pas en charge le contrôle d'accès name/password basé sur l'utilisateur.

WebSocket connexions simultanées et durée de connexion maximale

Le nombre de WebSocket connexions simultanées par instance de base de données Neptune est limité. Lorsque cette limite est atteinte, Neptune limite toute demande d'ouverture d'une nouvelle WebSocket connexion afin d'éviter d'utiliser toute la mémoire allouée.

Pour tous les types d'instances plus importants pris en charge par Neptune et toutes les instances sans serveur, le nombre maximal de WebSocket connexions simultanées est de 32 000 (32 768).

Le nombre maximal de WebSocket connexions simultanées pour les petits types d'instances est répertorié dans le tableau ci-dessous :

Type d'instance	Nombre maximum de WebSocket connexions simultanées
sans serveur	32768
db.r4.4xlarge	16384
db.r4.2xlarge	8192
db.r4.xlarge	4096
db.r4.large	2048
db.r5.4xlarge	16384
db.r5.2xlarge	8192
db.r5.xlarge	4096
db.r5.large	2048
db.r5d.8xlarge	32768
db.r5d.4xlarge	16384
db.r5d.2xlarge	8192

Type d'instance	Nombre maximum de WebSocket connexions simultanées
db.r5d.xlarge	4096
db.r5d.large	2048
db.t3.medium	512
db.t4g.medium	512
db.r6g.4xlarge	16384
db.r6g.2xlarge	8192
db.r6g.xlarge	4096
db.r6g.large	2048
db.r6g.4xlarge	16384
db.r6g.2xlarge	8192
db.r6gd.xlarge	4096
db.r6gd.large	2048
db.x2iezn.2xlarge	16384
db.x2iedn.xlarge	4096
db.x2gd.xlarge	4096
db.x2gd.large	2048
db.x2gd.4xlarge	16384
db.x2gd.2xlarge	8192
db.x2g.xlarge	4096
db.x2g.large	2048

Type d'instance	Nombre maximum de WebSocket connexions simultanées
db.x2g.4xlarge	16384
db.x2g.2xlarge	8192
db.x1e.xlarge	4096
db.x1e.2xlarge	16384
db.r6i.xlarge	4096
db.r6i.large	2048
db.r6i.2xlarge	8192
db.r7g.4xlarge	16384
db.r7g.2xlarge	8192
db.r7g.xlarge	4096
db.r7g.large	2048
db.r8g.4xlarge	16384
db.r8g.2xlarge	8192
db.r8g.xlarge	4096
db.r8g.large	2048
db.r7i.large	2048
db.r7i.xlarge	4096
db.r7i.2xlarge	8192

> **Note**
>
> Neptune ne prend plus en charge les types d'R4instances.

Lorsqu'un client ferme correctement une connexion, cette fermeture est immédiatement prise en compte dans le nombre de connexions ouvertes.

Si le client ne ferme pas la connexion, celle-ci peut être fermée automatiquement après un délai d'inactivité de 20 à 25 minutes (le délai d'inactivité est le temps écoulé depuis la réception du dernier message du client). Toutefois, tant que le délai d'inactivité n'est pas atteint, Neptune maintient la connexion ouverte indéfiniment.

Lorsque l'authentification IAM est activée, une WebSocket connexion est toujours déconnectée quelques minutes plus de 10 jours après son établissement, si elle n'a pas déjà été fermée d'ici là.

Limites applicables aux propriétés et aux étiquettes

Il n'y a pas de limite quant au nombre de sommets et d'arêtes, ou de quadruplets RDF que vous pouvez avoir dans un graphe.

Il n'y a pas non plus de limite quant au nombre de propriétés ou d'étiquettes qu'un sommet ou un arc peut avoir.

Une limite de taille de 55 Mo est appliquée à la taille d'une propriété ou d'une étiquette individuelle. En termes RDF, cela signifie que la valeur dans n'importe quelle colonne (S, P, O ou G) d'un quad RDF ne peut pas dépasser 55 Mo.

Si vous devez associer un objet plus grand, comme une image, à un sommet ou un nœud dans votre graphe, vous pouvez le stocker sous forme de fichier dans Amazon S3 et utiliser le chemin d'accès Amazon S3 comme propriété ou étiquette.

Limites qui affectent le chargeur en bloc Neptune

Vous ne pouvez pas mettre en file d'attente plus de 64 tâches de chargement en bloc Neptune à la fois.

Neptune ne conserve qu'une trace des 1 024 dernières tâches de chargement en bloc.

Neptune stocke uniquement les 10 000 dernières informations détaillées d'erreur par tâche.

Collaboration avec d'autres AWS services

Vous pouvez utiliser Amazon Neptune conjointement avec de nombreux autres AWS services :

Intégrations Neptune avec d'autres services

- AWS Glue : AWS Glue est un service d'intégration de données sans serveur qui vous aide à exécuter des tâches d'extraction, de transformation et de chargement (ETL) au niveau des données.

  Neptune fournit une bibliothèque open source qui simplifie l'utilisation de Python et de Gremlin dans le cadre d'une tâche Glue. neptune-python-utilities Le connecteur Neo4j Spark est également pris en charge pour exécuter les tâches Scala et openCypher Glue.
- Amazon SageMaker AI — Amazon SageMaker AI est une plateforme d'apprentissage automatique complète permettant de créer, de former et de déployer des modèles d'apprentissage automatique de haute qualité.

  Neptune s'intègre à l' SageMaker IA de deux manières principales :

  - Neptune fournit un package Python open source pour les blocs-notes Jupyter qui se trouve dans le projet Neptune Graph Notebook sur. GitHub Ce package contient un ensemble de magies Jupyter, des blocs-notes didactiques et des exemples de code qui fournissent un environnement de codage interactif dans lequel vous pouvez vous familiariser avec la technologie des graphes et Neptune. Neptune fournit un environnement entièrement géré pour les blocs-notes Jupyter hébergés par l' SageMaker IA et établit automatiquement des liens vers les blocs-notes dans le cadre du projet open source Neptune Graph Notebook.
  - La fonctionnalité Neptune ML permet de créer et d'entraîner des modèles de machine learning utiles sur des graphes de grande taille en quelques heures au lieu de plusieurs semaines. Pour ce faire, Neptune ML utilise la technologie GNN (Graph Neural Network) développée par Amazon SageMaker AI et la Deep Graph Library (DGL).
- AWS Lambda : les fonctions AWS Lambda ont de nombreuses utilisations dans les applications Neptune.

  Pour plus d'informations sur l'utilisation des fonctions Lambda avec l'un des pilotes et variantes de langage Gempda les plus courants, ainsi que pour des exemples spécifiques de fonctions Lambda écrites en Java JavaScript et Python, consultez. Utilisation AWS Lambda des fonctions dans Amazon Neptune

- Amazon Athena : Amazon Athena est un service de requêtes interactif qui facilite l'analyse des données dans Amazon Simple Storage Service et d'autres sources de données fédérées via la syntaxe SQL standard.

  Neptune fournit un connecteur à Athena qui permet à Athena de communiquer avec vos données stockées dans Neptune.
- AWS Database Migration Service (AWS DMS) — AWS Database Migration Service est un service AWS Web que vous pouvez utiliser pour migrer des données d'une base de données à une autre.

  AWS DMS peut charger des données dans Neptune à partir de bases de données sources prises en charge rapidement et en toute sécurité. La base de données source reste pleinement opérationnelle durant la migration, ce qui réduit au minimum les temps d'arrêt des applications qui l'utilisent.
- AWS Backup— AWS Backup est un service de sauvegarde entièrement géré qui facilite la centralisation et l'automatisation de la sauvegarde des données entre les AWS services dans le cloud et sur site.

  AWS Backup vous permet de créer des instantanés périodiques automatisés des clusters Neptune en utilisant votre politique de protection des données centralisée pour l'ensemble des services AWS pris en charge pour les bases de données, le stockage et le calcul.
- AWS SDK pour pandas — Le AWS SDK pour pandas (anciennement connu sous le nom de AWS Data Wrangler, or`awswrangler`) est une initiative python open source de services AWS professionnels qui étend la puissance de la bibliothèque d'analyse de données Python à plus de 30 services liés aux données, dont Neptune AWS, à la connexion `DataFrames` et à la connexion. `pandas` AWS

  Outre le kit SDK, il existe également un didacticiel expliquant comment l'utiliser avec Neptune, ainsi que plusieurs exemples de blocs-notes Neptune, à savoir la détection des réseaux de fraude, la détection des identités synthétiques et l'analyse logistique.
- Pilote JDBC : le pilote JDBC Neptune prend en charge les requêtes openCypher, Gremlin, SQL-Gremlin et SPARQL.

  La connectivité JDBC facilite la connexion à Neptune grâce à des outils de business intelligence (BI) tels que Tableau.

Outils et utilitaires Neptune

Amazon Neptune fournit divers outils et utilitaires qui contribuent à simplifier et à automatiser les tâches liées à un graphe. Parmi eux, citons :

Outils Amazon Neptune

- Utilitaire Amazon Neptune pour GraphQL — L'utilitaire Amazon Neptune pour GraphQL est un Node.js outil de ligne de commande open source qui peut vous aider à créer et à gérer une API GraphQL pour une base de données de graphes de propriétés Neptune. Il s'agit d'une méthode sans code permettant de créer un résolveur GraphQL pour les requêtes GraphQL qui ont un nombre variable de paramètres d'entrée et renvoient un nombre variable de champs imbriqués.
- Nodestream — Nodestream est un framework permettant de modéliser sémantiquement des données sous forme de graphe. Il est conçu pour être flexible et extensible, vous permettant de définir la manière dont les données sont collectées et modélisées sous forme de graphique. Il utilise une approche basée sur le pipeline pour définir la manière dont les données sont collectées et traitées, et il fournit un moyen de définir la manière dont le graphique doit être mis à jour lorsque le schéma change.
- Serveurs de requête Amazon Neptune MCP : serveur MCP (Model Context Protocol) pour Amazon Neptune qui prend en charge à la fois Neptune Database et Neptune Analytics en permettant d'exécuter des requêtes graphiques (OpenCypher et G705) ainsi que de récupérer le schéma du graphe.
- Serveurs de mémoire Amazon Neptune MCP : serveur MCP (Model Context Protocol) pour Amazon Neptune qui fournit de la mémoire aux agents, stockée dans un graphe de connaissances par rapport à Amazon Neptune.

Utilitaire Amazon Neptune pour GraphQL

L'utilitaire Amazon Neptune pour GraphQL est un outil de Node.js ligne de commande open source qui peut vous aider à créer et à gérer une API GraphQL pour une base de données de graphes de propriétés Neptune (il ne fonctionne pas encore avec les données RDF). Il s'agit d'une méthode sans code permettant de créer un résolveur GraphQL pour les requêtes GraphQL qui ont un nombre variable de paramètres d'entrée et renvoient un nombre variable de champs imbriqués.

Il a été publié en tant que projet open source situé à l'adresse https://github.com/aws/amazon-neptune-for-graphql.

Vous pouvez installer cet utilitaire à l'aide de NPM comme suit (voir Installation et configuration pour plus d'informations) :

```
npm i @aws/neptune-for-graphql -g
```

L'utilitaire peut découvrir le schéma d'un graphe de propriétés Neptune existant, y compris les nœuds, les arêtes, les propriétés et la cardinalité des arêtes. Il génère ensuite un schéma GraphQL avec les directives nécessaires pour mapper les types GraphQL avec les nœuds et les arêtes de la base de données, puis génère automatiquement le code du résolveur. Le code du résolveur est conçu pour minimiser la latence en renvoyant uniquement les données demandées par la requête GraphQL.

Vous pouvez également commencer avec un schéma GraphQL existant et une base de données Neptune vide, puis laisser l'utilitaire déduire les directives nécessaires pour mapper ce schéma GraphQL avec les nœuds et arêtes des données à charger dans la base de données. Vous pouvez également commencer par un schéma GraphQL et des directives que vous avez déjà créés ou modifiés.

L'utilitaire est capable de créer toutes les AWS ressources dont il a besoin pour son pipeline, notamment l' AWS AppSync API, les rôles IAM, la source de données, le schéma et le résolveur, ainsi que la fonction AWS Lambda qui interroge Neptune.

> **Note**
>
> Command-line les exemples ici supposent une console Linux. Si vous utilisez Windows, remplacez les barres obliques inverses (« \ ») à la fin des lignes par des accents circonflexes (« ^ »).

Rubriques

- Installation et configuration de l'utilitaire Amazon Neptune pour GraphQL
- Numérisation de données dans une base de données Neptune existante
- À partir d'un schéma GraphQL sans directives
- Utilisation des directives pour un schéma GraphQL
- Command-line arguments pour l'utilitaire GraphQL

Installation et configuration de l'utilitaire Amazon Neptune pour GraphQL

Si vous comptez utiliser l'utilitaire avec une base de données Neptune existante, il doit pouvoir se connecter au point de terminaison de la base de données. Par défaut, une base de données Neptune n'est accessible que depuis le VPC où elle se trouve.

L'utilitaire étant un outil de Node.js ligne de commande, vous devez l'avoir installé Node.js (version 18 ou supérieure) pour qu'il puisse s'exécuter. Pour effectuer l'installation Node.js sur une instance EC2 dans le même VPC que votre base de données Neptune, suivez les instructions ici. La taille minimale de l'instance pour exécuter l'utilitaire est t2.micro. Lors de la création de l'instance, sélectionnez le VPC de base de données Neptune dans le menu déroulant Groupes de sécurité courants.

Pour installer l'utilitaire lui-même sur une instance EC2 ou sur votre ordinateur local, utilisez NPM :

```
npm i @aws/neptune-for-graphql -g
```

Vous pouvez ensuite exécuter la commande d'aide de l'utilitaire pour vérifier s'il est correctement installé :

```
neptune-for-graphql --help
```

Vous pouvez également installer le AWS CLI pour gérer les AWS ressources.

Numérisation de données dans une base de données Neptune existante

Que vous connaissiez GraphQL ou non, la commande ci-dessous est le moyen le plus rapide de créer une API GraphQL. Cela suppose que vous avez installé et configuré l'utilitaire Neptune pour GraphQL comme décrit dans la section d'installation, afin qu'il soit connecté au point de terminaison de votre base de données Neptune.

```
neptune-for-graphql \
  --input-graphdb-schema-neptune-endpoint (your neptune database endpoint):(port
 number) \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (your new GraphQL API name) \
  --output-resolver-query-https
```

L'utilitaire analyse la base de données pour découvrir le schéma des nœuds, des arêtes et des propriétés qu'elle contient. Sur la base de ce schéma, il déduit un schéma GraphQL avec les

requêtes et les mutations associées. Il crée ensuite une API AppSync GraphQL et les AWS ressources nécessaires pour l'utiliser. Ces ressources incluent une paire de rôles IAM et une fonction Lambda contenant le code du résolveur GraphQL.

Une fois l'utilitaire terminé, vous trouverez une nouvelle API GraphQL dans la AppSync console sous le nom que vous avez attribué dans la commande. Pour le tester, utilisez l'option AppSync Requêtes du menu.

Si vous réexécutez la même commande après avoir ajouté des données supplémentaires à la base de données, l' AppSync API et le code Lambda seront mis à jour en conséquence.

Pour libérer toutes les ressources associées à la commande, exécutez :

```
neptune-for-graphql \
  --remove-aws-pipeline-name (your new GraphQL API name from above)
```

À partir d'un schéma GraphQL sans directives

Vous pouvez commencer avec une base de données Neptune vide et utiliser un schéma GraphQL sans directives pour créer les données et les interroger. La commande ci-dessous crée automatiquement les ressources AWS nécessaires à cette fin :

```
neptune-for-graphql \
  --input-schema-file (your GraphQL schema file)
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (name for your new GraphQL API) \
  --create-update-aws-pipeline-neptune-endpoint (your Neptune database endpoint):(port
 number) \
  --output-resolver-query-https
```

Le fichier de schéma GraphQL doit inclure les types de schéma GraphQL, comme indiqué dans l'exemple TODO ci-dessous. L'utilitaire analyse le schéma et crée une version étendue en fonction de vos types. Il ajoute des requêtes et des mutations pour les nœuds stockés dans la base de données orientée graphe, et si votre schéma comporte des types imbriqués, il ajoute des relations entre les types stockés sous forme d'arêtes dans la base de données.

L'utilitaire crée une API AppSync GraphQL et toutes les AWS ressources nécessaires. Ces ressources incluent une paire de rôles IAM et une fonction Lambda contenant le code du résolveur GraphQL. Lorsque la commande est terminée, vous pouvez trouver une nouvelle API GraphQL

portant le nom que vous avez spécifié dans la AppSync console. Pour le tester, utilisez Requêtes dans le AppSync menu.

L'exemple suivant illustre comment cela fonctionne :

Exemple Todo avec un schéma GraphQL sans directives

Dans cet exemple, nous partons d'un schéma Todo GraphQL sans directives, que vous pouvez trouver dans *???samples???* le répertoire. Il inclut les deux types suivants :

```
type Todo {
  name: String
  description: String
  priority: Int
  status: String
  comments: [Comment]
}

type Comment {
  content: String
}
```

Cette commande traite le schéma Todo et un point de terminaison d'une base de données Neptune vide pour créer une API GraphQL dans : AWS AppSync

```
neptune-for-graphql /
  --input-schema-file ./samples/todo.schema.graphql \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name TodoExample \
  --create-update-aws-pipeline-neptune-endpoint (empty Neptune database endpoint):(port
 number) \
  --output-resolver-query-https
```

L'utilitaire crée un nouveau fichier dans le dossier de sortie appeléTodoExample.source.graphql, et l'API GraphQL dans. AppSync L'utilitaire en déduit ce qui suit :

- Dans le type Todo, il a été ajouté `@relationship` pour un nouveau CommentEdge type. Cela indique au résolveur de connecter Todo à Comment à l'aide d'un edge de base de données de graphes appelé. CommentEdge

- Il a ajouté une nouvelle entrée appelée TodoInput pour aider les requêtes et les mutations.
- Il a ajouté deux requêtes pour chaque type (Todo, Comment) : l'une pour récupérer un seul type en utilisant un `id` ou l'un des champs de type répertoriés dans l'entrée, et l'autre pour récupérer plusieurs valeurs, filtrées à l'aide de l'entrée correspondant à ce type.
- Il a ajouté trois mutations pour chaque type : créer, mettre à jour et supprimer. Le type à supprimer est spécifié à l'aide d'un `id` ou de l'entrée correspondant à ce type. Ces mutations concernent les données stockées dans la base de données Neptune.
- Il a ajouté deux mutations pour les connexions : connecter et supprimer. Elles utilisent en entrée les ID de nœud des sommets de départ et de destination utilisés par Neptune, et les connexions sont des arêtes dans la base de données.

Le résolveur reconnaît les requêtes et les mutations par leur nom, mais vous pouvez les personnaliser comme indiqué ci-dessous.

Voici le contenu du fichier `TodoExample.source.graphql` :

```
type Todo {
  _id: ID! @id
  name: String
  description: String
  priority: Int
  status: String
  comments(filter: CommentInput, options: Options): [Comment] @relationship(type:
 "CommentEdge", direction: OUT)
  bestComment: Comment @relationship(type: "CommentEdge", direction: OUT)
  commentEdge: CommentEdge
}

type Comment {
  _id: ID! @id
  content: String
}

input Options {
  limit: Int
}

input TodoInput {
  _id: ID @id
  name: String
```

```
  description: String
  priority: Int
  status: String
}

type CommentEdge {
  _id: ID! @id
}

input CommentInput {
  _id: ID @id
  content: String
}

input Options {
  limit: Int
}

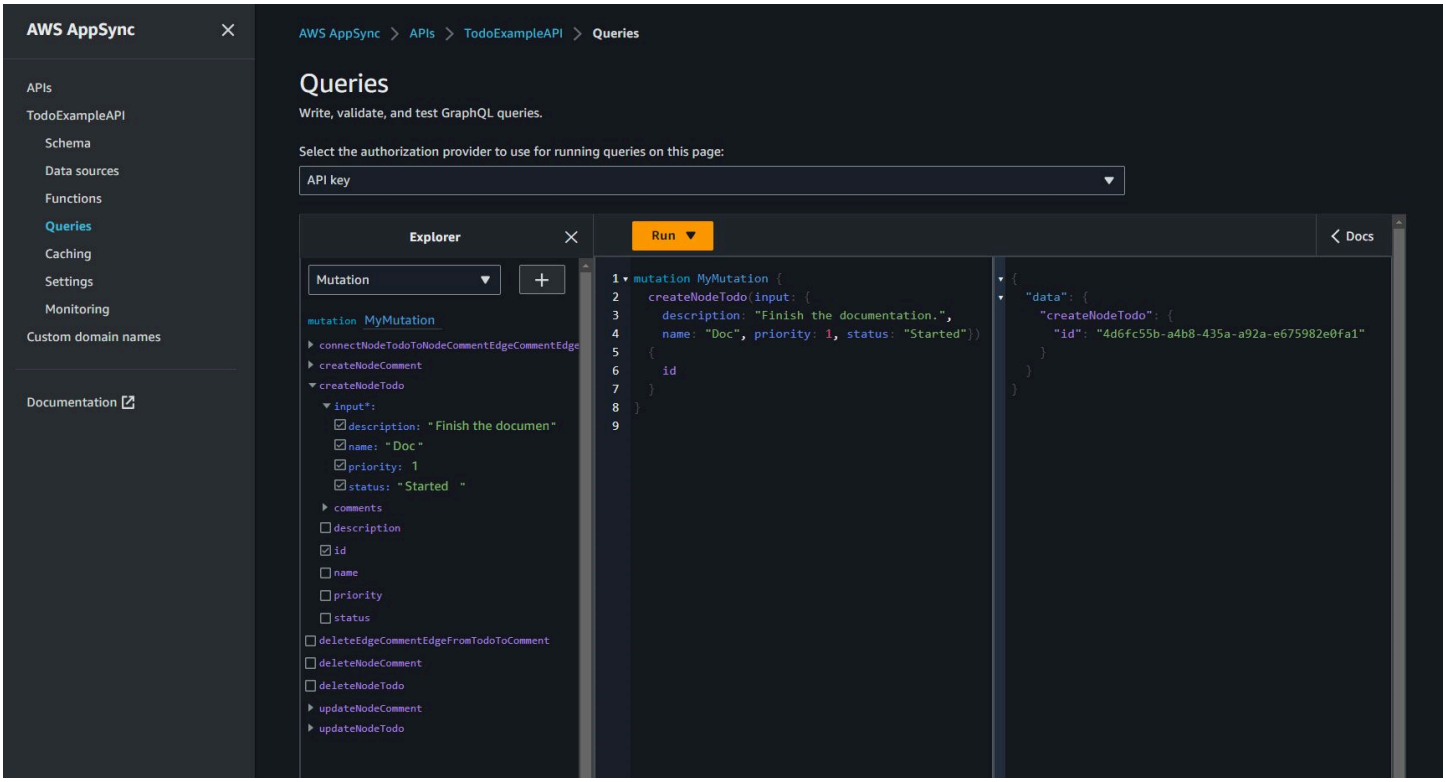
type Query {
  getNodeTodo(filter: TodoInput, options: Options): Todo
  getNodeTodos(filter: TodoInput): [Todo]
  getNodeComment(filter: CommentInput, options: Options): Comment
  getNodeComments(filter: CommentInput): [Comment]
}

type Mutation {
  createNodeTodo(input: TodoInput!): Todo
  updateNodeTodo(input: TodoInput!): Todo
  deleteNodeTodo(_id: ID!): Boolean
  connectNodeTodoToNodeCommentEdgeCommentEdge(from_id: ID!, to_id: ID!): CommentEdge
  deleteEdgeCommentEdgeFromTodoToComment(from_id: ID!, to_id: ID!): Boolean
  createNodeComment(input: CommentInput!): Comment
  updateNodeComment(input: CommentInput!): Comment
  deleteNodeComment(_id: ID!): Boolean
}

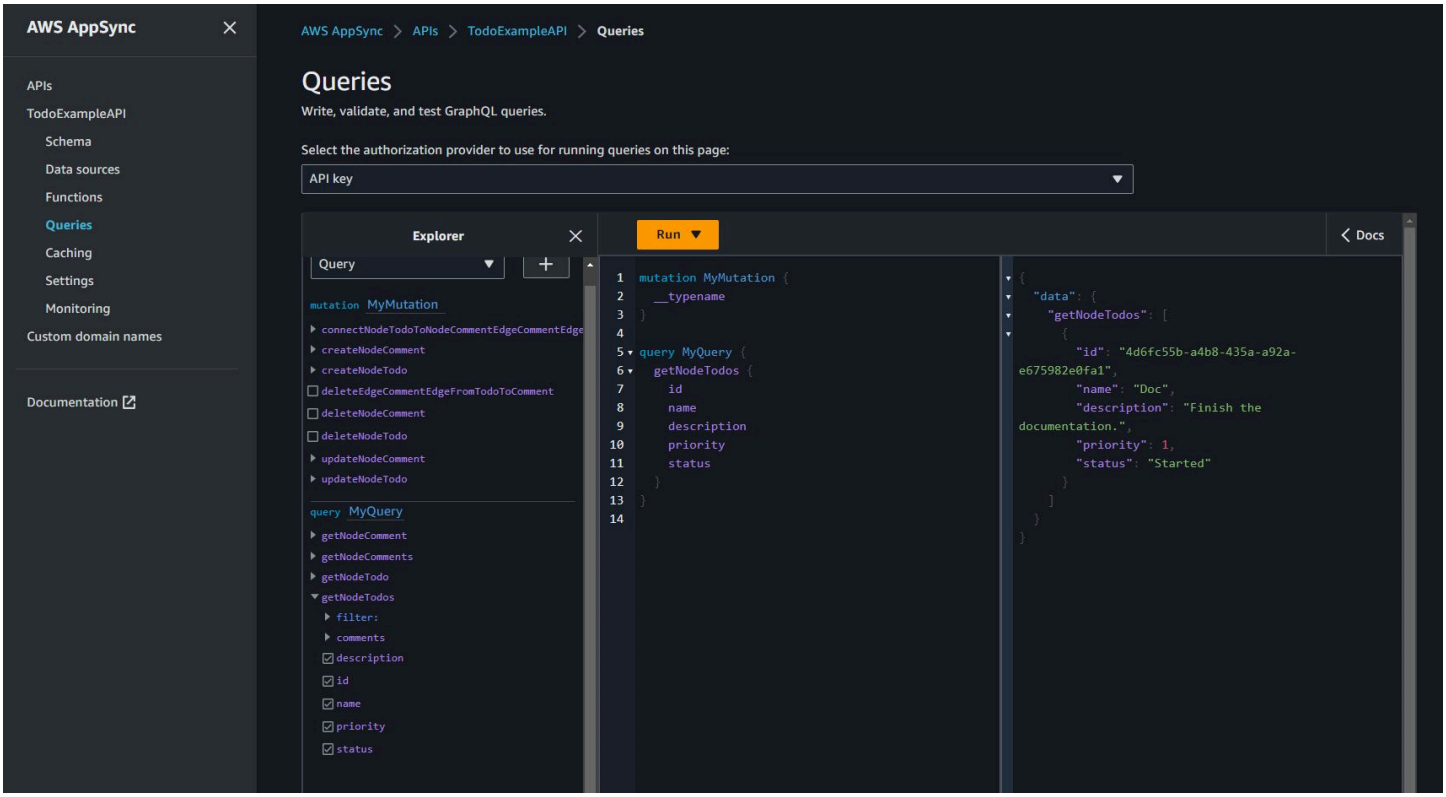
schema {
  query: Query
  mutation: Mutation
}
```

Vous pouvez désormais créer et interroger des données. Voici un aperçu de la console AppSync Queries utilisée pour tester la nouvelle API GraphQL, nommée `TodoExampleAPI` dans ce cas. Dans

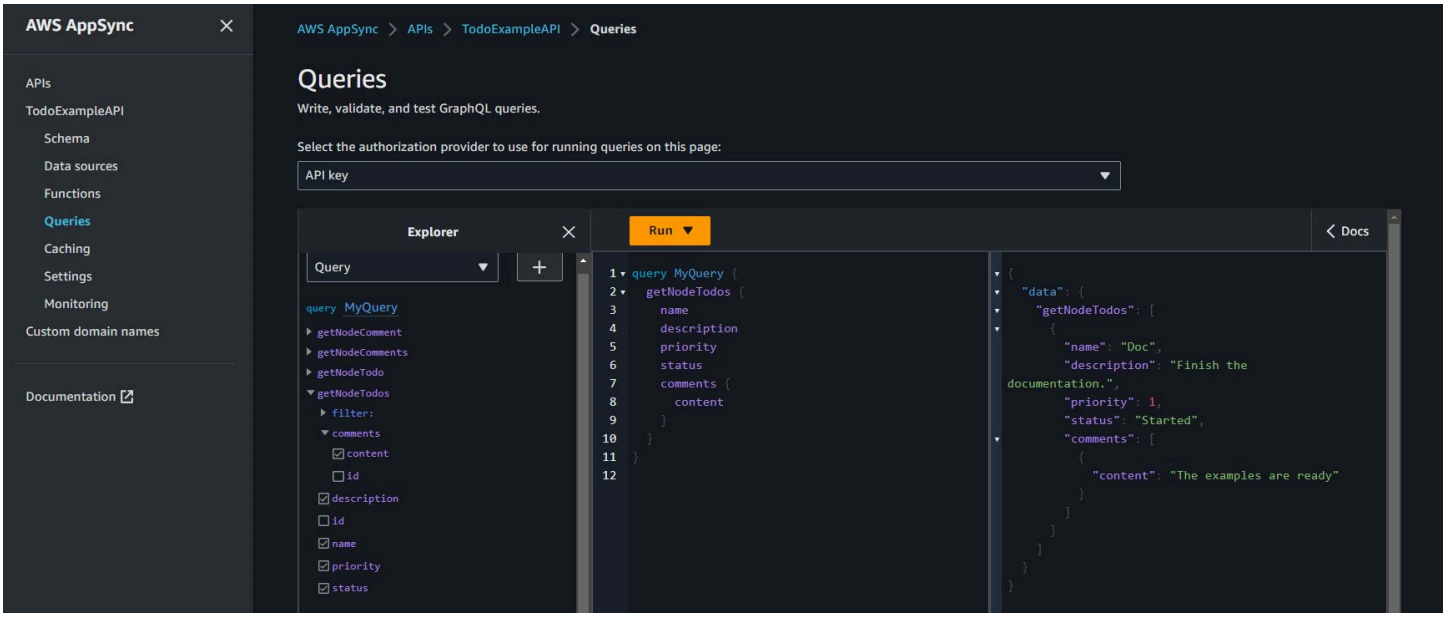
la fenêtre du milieu, l'explorateur affiche une liste de requêtes et de mutations parmi lesquelles vous pouvez sélectionner une requête, les paramètres d'entrée et les champs de retour. Cette capture d'écran montre la création d'un type de nœud Todo à l'aide de la mutation `createNodeTodo` :



Cette capture d'écran montre comment interroger tous les nœuds Todo à l'aide de la requête `getNodeTodos` :



Après avoir créé un commentaire à l'aide de `createNodeComment`, vous pouvez utiliser la mutation `connectNodeTodoToNodeCommentEdgeCommentEdge` pour les connecter en spécifiant leur ID. Voici une requête imbriquée permettant de récupérer les nœuds Todo et les commentaires qui y sont associés :



Si vous souhaitez apporter des modifications au fichier `TodoExample.source.graphql` comme décrit dans la section Utilisation des directives, vous pouvez utiliser le schéma modifié comme entrée et exécuter à nouveau l'utilitaire. L'utilitaire modifiera ensuite l'API GraphQL en conséquence.

Utilisation des directives pour un schéma GraphQL

Vous pouvez partir d'un schéma GraphQL qui contient déjà des directives, en utilisant une commande comme celle-ci :

```
neptune-for-graphql \
  --input-schema-file (your GraphQL schema file with directives) \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (name for your new GraphQL API) \
  --create-update-aws-pipeline-neptune-endpoint  (empty Neptune database
 endpoint):(port number) \
  --output-resolver-query-https
```

Vous pouvez modifier les directives créées par l'utilitaire ou ajouter les vôtres à un schéma GraphQL. Voici quelques méthodes d'utilisation des directives :

Exécutation de l'utilitaire pour qu'il ne génère pas de mutations

Pour empêcher l'utilitaire de générer des mutations dans l'API GraphQL, utilisez l'option `--output-schema-no-mutations` de la commande `neptune-for-graphql`.

La directive `@alias`

La directive `@alias` peut être appliquée aux types de schéma ou aux champs GraphQL. Elle mappe différents noms entre la base de données orientée graphe et le schéma GraphQL. La syntaxe est la suivante :

```
@alias(property: (property name))
```

Dans l'exemple ci-dessous, `airport` est l'étiquette du nœud de base de données orientée graphe mappée avec le type GraphQL `Airport`, et `desc` est la propriété du nœud de graphe mappée avec le champ `description` (voir l'exemple des routes aériennes) :

```
type Airport @alias(property: "airport") {
  city: String
  description: String @alias(property: "desc")
```

```
}
```

Notez que le formatage GraphQL standard nécessite des noms de Pascal-casing types et des noms de champs en forme de chameau.

La directive `@relationship`

La directive `@relationship` mappe les types GraphQL imbriqués avec les arêtes de la base de données orientée graphe. La syntaxe est la suivante :

```
@relationship(edgeType: (edge name), direction: (IN or OUT))
```

Voici un exemple de commande :

```
type Airport @alias(property: "airport") {
  ...
  continentContainsIn: Continent @relationship(edgeType: "contains", direction: IN)
  countryContainsIn: Country @relationship(edgeType: "contains", direction: IN)
  airportRoutesOut(filter: AirportInput, options: Options): [Airport]
 @relationship(edgeType: "route", direction: OUT)
  airportRoutesIn(filter: AirportInput, options: Options): [Airport]
 @relationship(edgeType: "route", direction: IN)
}
```

Vous trouverez des directives `@relationship` à la fois dans l'exemple Todo et dans l'exemple des routes aériennes.

Les directives `@graphQuery` et `@cypher`

Vous pouvez définir des requêtes openCypher pour résoudre une valeur de champ, ajouter des requêtes ou ajouter des mutations. Par exemple, cela ajoute un nouveau champ `outboundRoutesCount` au type `Airport` pour compter les routes sortantes :

```
type Airport @alias(property: "airport") {
  ...
  outboundRoutesCount: Int @graphQuery(statement: "MATCH (this)-[r:route]->(a) RETURN
 count(r)")
}
```

Voici un exemple de nouvelles requêtes et mutations :

```
type Query {
  getAirportConnection(fromCode: String!, toCode: String!): Airport \
    @cypher(statement: \
      "MATCH (:airport{code: '$fromCode'})-[:route]->(this:airport)-[:route]-
>(:airport{code:'$toCode'})")
}

type Mutation {
  createAirport(input: AirportInput!): Airport @graphQuery(statement: "CREATE
 (this:airport {$input}) RETURN this")
  addRoute(fromAirportCode:String, toAirportCode:String, dist:Int): Route \
    @graphQuery(statement: \
     "MATCH (from:airport{code:'$fromAirportCode'}),
 (to:airport{code:'$toAirportCode'}) \
      CREATE (from)-[this:route{dist:$dist}]->(to) \
      RETURN this")
}
```

Notez que si vous omettez RETURN, le résolveur suppose que le mot clé `this` est la portée renvoyée.

Vous pouvez également ajouter une requête ou une mututation à l'aide d'une requête Gremlin :

```
type Query {
  getAirportWithGremlin(code:String): Airport \
    @graphQuery(statement: "g.V().has('airport', 'code', '$code').elementMap()")  #
 single node
  getAirportsWithGremlin: [Airport] \
    @graphQuery(statement: "g.V().hasLabel('airport').elementMap().fold()")       #
 list of nodes
  getCountriesCount: Int \
    @graphQuery(statement: "g.V().hasLabel('country').count()")                   #
 scalar
}
```

À l'heure actuelle, les requêtes Gremlin sont limitées à celles qui renvoient des valeurs scalaires, `elementMap()` pour un seul nœud ou `elementMap().fold()` pour une liste de nœuds.

La directive **`@id`**

La directive `@id` identifie le champ mappé avec l'entité de base de données orientée graphe `id`. Les bases de données orientées graphe telles qu'Amazon Neptune disposent toujours d'un `id`

unique pour les nœuds et les arêtes. Celui-ci est attribué lors des importations en bloc ou est généré automatiquement. Par exemple :

```
type Airport {
  _id: ID! @id
  city: String
  code: String
}
```

Noms de type, de requête et de mutation réservés

L'utilitaire génère automatiquement des requêtes et des mutations pour créer une API GraphQL fonctionnelle. Le modèle de ces noms est reconnu par le résolveur et est réservé. Voici des exemples pour le type `Airport` et le type de connexion `Route` :

Le nom `Options` est réservé.

```
input Options {
  limit: Int
}
```

Les paramètres de fonction `filter` et `options` sont réservés.

```
type Query {
  getNodeAirports(filter: AirportInput, options: Options): [Airport]
}
```

Le préfixe getNode des noms de requêtes est réservé, et les préfixes des noms de mutations tels que `createNode`, `updateNode`, `deleteNode`, `connectNode`, `deleteNode`, `updateEdge` et `deleteEdge` sont réservés.

```
type Query {
  getNodeAirport(id: ID, filter: AirportInput): Airport
  getNodeAirports(filter: AirportInput): [Airport]
}

type Mutation {
  createNodeAirport(input: AirportInput!): Airport
  updateNodeAirport(id: ID!, input: AirportInput!): Airport
  deleteNodeAirport(id: ID!): Boolean
```

```
  connectNodeAirportToNodeAirportEdgeRout(from: ID!, to: ID!, edge: RouteInput!): Route
  updateEdgeRouteFromAirportToAirport(from: ID!, to: ID!, edge: RouteInput!): Route
  deleteEdgeRouteFromAirportToAirport(from: ID!, to: ID!): Boolean
}
```

Application de modifications au schéma GraphQL

Vous pouvez modifier le schéma source GraphQL et réexécuter l'utilitaire pour obtenir le dernier schéma de la base de données Neptune. Chaque fois que l'utilitaire découvre un nouveau schéma dans la base de données, il génère un nouveau schéma GraphQL.

Vous pouvez également modifier manuellement le schéma source GraphQL et réexécuter l'utilitaire en utilisant le schéma source comme entrée au lieu du point de terminaison de la base de données Neptune.

Enfin, vous pouvez placer vos modifications dans un fichier à l'aide de ce format JSON :

```
[
  {
    "type": "(GraphQL type name)",
    "field": "(GraphQL field name)",
    "action": "(remove or add)",
    "value": "(value)"
  }
]
```

Par exemple :

```
[
  {
    "type": "Airport",
    "field": "outboundRoutesCountAdd",
    "action": "add",
    "value":"outboundRoutesCountAdd: Int @graphQuery(statement: \"MATCH (this)-
[r:route]->(a) RETURN count(r)\")"
  },
  {
    "type": "Mutation",
    "field": "deleteNodeVersion",
    "action": "remove",
    "value": ""
  },
```

```
  {
    "type": "Mutation",
    "field": "createNodeVersion",
    "action": "remove",
    "value": ""
  }
]
```

Ensuite, lorsque vous exécutez l'utilitaire sur ce fichier à l'aide du paramètre `--input-schema-changes-file` de la commande, l'utilitaire applique les modifications immédiatement.

Command-line arguments pour l'utilitaire GraphQL

- **`--help, -h`** : renvoie le texte d'aide de l'utilitaire GraphQL à la console.

- **`--input-schema`** ***`(schema text)`*** : schéma GraphQL, avec ou sans directives, à utiliser en entrée.

- **`--input-schema-file`** ***`(file URL)`*** : URL d'un fichier contenant un schéma GraphQL à utiliser en entrée.

- **`--input-schema-changes-file`** ***`(file URL)`*** : URL d'un fichier contenant les modifications que vous souhaitez apporter à un schéma GraphQL. Si vous exécutez l'utilitaire sur une base de données Neptune à plusieurs reprises et que vous modifiez également manuellement le schéma source GraphQL, par exemple en ajoutant une requête personnalisée, les modifications manuelles seront perdues. Pour éviter cela, placez les modifications dans un fichier de modifications et transmettez-le à l'aide de cet argument.

  Le fichier de modifications utilise le format JSON suivant :

```
[
  {
    "type": "(GraphQL type name)",
    "field": "(GraphQL field name)",
    "action": "(remove or add)",
    "value": "(value)"
  }
```

```
]
```

Pour plus d'informations, consultez l'exemple Todo.

- **`--input-graphdb-schema`** ***`(schema text)`*** : au lieu d'exécuter l'utilitaire sur une base de données Neptune, vous pouvez exprimer un schéma graphdb sous forme de texte à utiliser comme entrée. Un schéma graphdb a un format JSON comme celui-ci :

```
{
  "nodeStructures": [
    { "label":"nodelabel1",
      "properties": [
        { "name":"name1", "type":"type1" }
      ]
    },
    { "label":"nodelabel2",
      "properties": [
          { "name":"name2", "type":"type1" }
      ]
    }
  ],
  "edgeStructures": [
    {
      "label":"label1",
      "directions": [
        { "from":"nodelabel1", "to":"nodelabel2", "relationship":"ONE-ONE|ONE-MANY|
MANY-MANY"  }
      ],
      "properties": [
        { "name":"name1", "type":"type1" }
      ]
    }
  ]
}
```

- **`--input-graphdb-schema-file`** ***`(file URL)`*** : au lieu d'exécuter l'utilitaire sur une base de données Neptune, vous pouvez enregistrer un schéma graphdb dans un fichier à utiliser comme entrée. Consultez `--input-graphdb-schema` ci-dessus pour voir un exemple du format JSON pour un fichier de schéma graphdb.

- **`--input-graphdb-schema-neptune-endpoint` *`(endpoint URL)`*** : point de terminaison de base de données Neptune à partir duquel l'utilitaire doit extraire le schéma graphdb.

- **`--output-schema-file` *`(file name)`*** : nom du fichier de sortie pour le schéma GraphQL. S'il n'est pas spécifié, la valeur par défaut est `output.schema.graphql`, sauf si un nom de pipeline a été défini avec `--create-update-aws-pipeline-name`, auquel cas le nom de fichier par défaut est *`(pipline name)`*`.schema.graphql`.

- **`--output-source-schema-file` *`(file name)`*** : nom du fichier de sortie correspondant au schéma GraphQL avec directives. S'il n'est pas spécifié, la valeur par défaut est `output.source.schema.graphql`, sauf si un nom de pipeline a été défini avec `--create-update-aws-pipeline-name`, auquel cas le nom par défaut est *`(pipeline name)`*`.source.schema.graphql`.

- **`--output-schema-no-mutations`** : si cet argument est présent, l'utilitaire ne génère aucune mutation dans l'API GraphQL, uniquement des requêtes.

- **`--output-neptune-schema-file` *`(file name)`*** : nom du fichier de sortie du schéma Neptune graphdb découvert par l'utilitaire. S'il n'est pas spécifié, la valeur par défaut est `output.graphdb.json`, sauf si un nom de pipeline a été défini avec `--create-update-aws-pipeline-name`, auquel cas le nom de fichier par défaut est *`(pipeline name)`*`.graphdb.json`.

- **`--output-js-resolver-file` *`(file name)`*** : nom du fichier de sortie correspondant à une version du code du résolveur. S'il n'est pas spécifié, la valeur par défaut est `output.resolver.graphql.js`, sauf si un nom de pipeline a été défini avec `--create-update-aws-pipeline-name`, auquel cas le nom de fichier est *`(pipeline name)`*`.resolver.graphql.js`.

Ce fichier est compressé dans le package de code chargé vers la fonction Lambda qui exécutera le résolveur.

- **`--output-resolver-query-sdk`** : cet argument indique que la fonction Lambda de l'utilitaire doit interroger Neptune à l'aide du kit SDK de données Neptune, qui est disponible à partir de la version 1.2.1.0.R5 du moteur Neptune (il s'agit de la version par défaut). Toutefois, si l'utilitaire détecte une ancienne version du moteur Neptune, il suggère d'utiliser plutôt l'option Lambda HTTPS, que vous pouvez invoquer à l'aide de l'argument `--output-resolver-query-https`.

- **`--output-resolver-query-https`** : cet argument indique que la fonction Lambda de l'utilitaire doit interroger Neptune à l'aide de l'API Neptune HTTPS.

- **`--create-update-aws-pipeline`**— Cet argument déclenche la création des AWS ressources que l'API GraphQL doit utiliser, notamment l'API AppSync GraphQL et le Lambda qui exécute le résolveur.

- **`--create-update-aws-pipeline-name`** ***`(pipeline name)`***— Cet argument définit le nom du pipeline, comme l'`pipeline-name`API AppSync ou la fonction de la `pipeline-name` fonction Lambda. Si aucun nom n'est spécifié, `--create-update-aws-pipeline` utilise le nom de la base de données `Neptune` .
- **`--create-update-aws-pipeline-region`** ***`(AWS region)`***— Cet argument définit la AWS région dans laquelle le pipeline de l'API GraphQL est créé. Si elle n'est pas spécifiée, la région par défaut est `us-east-1` ou la région où se trouve la base de données Neptune, extraite du point de terminaison de la base de données.
- **`--create-update-aws-pipeline-neptune-endpoint`** ***`(endpoint URL)`*** : cet argument définit le point de terminaison de la base de données Neptune utilisé par la fonction Lambda pour interroger la base de données. S'il n'est pas défini, le point de terminaison défini par `--input-graphdb-schema-neptune-endpoint` est utilisé.
- **`--remove-aws-pipeline-name`** ***`(pipeline name)`*** : cet argument supprime un pipeline créé à l'aide de `--create-update-aws-pipeline`. Les ressources à supprimer sont répertoriées dans un fichier nommé ***`(pipeline name)`***`.resources.json`.

- **`--output-aws-pipeline-cdk`**— Cet argument déclenche la création d'un fichier CDK qui peut être utilisé pour créer les AWS ressources de l'API GraphQL, notamment l'API GraphQL et la AppSync fonction Lambda qui exécute le résolveur.
- **`--output-aws-pipeline-cdk-neptume-endpoint`** ***`(endpoint URL)`*** : cet argument définit le point de terminaison de la base de données Neptune utilisé par la fonction Lambda pour interroger la base de données Neptune. S'il n'est pas défini, le point de terminaison défini par `--input-graphdb-schema-neptune-endpoint` est utilisé.
- **`--output-aws-pipeline-cdk-name`** ***`(pipeline name)`***— Cet argument définit le nom du pipeline pour l' AppSync API et la fonction Lambda pipeline-name à utiliser. S'il n'est pas spécifié, `--create-update-aws-pipeline` utilise le nom de la base de données Neptune.
- **`--output-aws-pipeline-cdk-region`** ***`(AWS region)`***— Cela définit la AWS région dans laquelle le pipeline de l'API GraphQL est créé. Si elle n'est pas spécifiée, la région par défaut est `us-east-1` ou la région où se trouve la base de données Neptune, extraite du point de terminaison de la base de données.
- **`--output-aws-pipeline-cdk-file`** ***`(file name)`*** : définit le nom du fichier CDK. Si aucune valeur n'est définie, la valeur par défaut est *`(pipeline name)`*`-cdk.js`.

Flux de nœuds

Nodestream est un framework permettant de modéliser sémantiquement des données sous forme de graphe. Il est conçu pour être flexible et extensible, vous permettant de définir la manière dont les données sont collectées et modélisées sous forme de graphique. Il utilise une approche basée sur le pipeline pour définir la manière dont les données sont collectées et traitées, et il fournit un moyen de définir la manière dont le graphique doit être mis à jour lorsque le schéma change. Tout cela se fait à l'aide d'un fichier de configuration simple et lisible par l'homme au format yaml. Pour ce faire, Nodestream utilise un certain nombre de concepts de base, notamment les pipelines, les extracteurs, les transformateurs, les filtres, les interpréteurs, les interprétations et les migrations.

À partir de Nodestream 0.12, Amazon Neptune est pris en charge à la fois pour Neptune Database et Neptune Analytics.

Consultez la documentation Nodestream pour savoir comment configurer et utiliser Nodestream avec Neptune : support Nodestream pour Amazon Neptune.

Nodestream with Neptune prend actuellement en charge les pipelines ETL standard ainsi que les pipelines Time to Live (TTL). Les pipelines ETL permettent l'ingestion de données en masse dans

Neptune à partir d'un éventail de sources de données et de formats beaucoup plus large que ce qui était possible auparavant dans Neptune, notamment :

- Nomenclature logicielle
- Fichiers tels que CSV, JSON, JSONL, Parquet, txt et yaml
- Kafka
- Athéna
- API REST

Nodestream prend entièrement en charge l'authentification IAM lors de la connexion à Amazon Neptune, à condition que les informations d'identification soient correctement configurées. Consultez le guide des informations d'identification boto3 pour plus d'informations sur la configuration correcte des informations d'identification.

Le mécanisme TTL de Nodestream permet également de nouvelles fonctionnalités qui n'étaient pas disponibles auparavant dans Neptune. En annotant les éléments graphiques ingérés avec des horodatages, Nodestream peut créer des pipelines qui expirent automatiquement et suppriment les données dont la durée de vie est dépassée.

Erreurs de service Neptune

Amazon Neptune comporte deux ensembles d'erreurs différents :

- Les erreurs de moteur de graphe qui sont spécifiques aux points de terminaison de cluster de bases de données
- Les erreurs associées à la création et à la modification APIs des ressources Neptune à l'aide du AWS SDK et AWS Command Line Interface ().AWS CLI

Rubriques

- Instances de base de données disponibles insuffisantes
- Messages et codes d'erreur du moteur de graphe
- Messages et codes d'erreur liés à l'API de gestion de cluster de base de données
- Message d'erreur et de flux liés au chargeur Neptune

Instances de base de données disponibles insuffisantes

L'erreur `InsufficientDBInstanceCapacity` peut être renvoyée lorsque vous essayez de créer, de démarrer ou de modifier une instance de base de données. Elle peut également être renvoyée lorsque vous essayez de restaurer une instance de base de données à partir d'un instantané de base de données. Lorsque cette erreur est renvoyée, la cause la plus fréquente est que la classe d'instance de base de données spécifique n'est pas disponible dans la zone de disponibilité demandée. Vous pouvez essayer une des actions suivantes pour résoudre le problème :

- Renouveler la demande avec une classe d'instance de base de données différente.
- Renouveler la demande avec une zone de disponibilité différente.
- Renouveler la demande sans spécifier de zone de disponibilité explicite.

Pour plus d'informations sur la résolution des problèmes de capacité d'instance pour Amazon EC2, consultez la section Capacité d'instance insuffisante dans le guide de EC2 l'utilisateur Amazon.

Messages et codes d'erreur du moteur de graphe

Les points de terminaison Amazon Neptune renvoient les erreurs standard pour Gremlin et SPARQL lorsqu'elles sont détectées.

Des erreurs spécifiques à Neptune peuvent également être renvoyées depuis les mêmes points de terminaison. Cette section documente les messages d'erreur Neptune, les codes et les actions recommandées.

> **Note**
>
> Ces erreurs concernent uniquement les points de terminaison de cluster de bases de données Neptune. APIs Pour créer et modifier des ressources Neptune avec le AWS SDK et AWS CLI présentant un ensemble différent d'erreurs courantes. Pour plus d'informations sur ces erreurs, consultez the section called “Erreurs d'API”.

Format des erreurs du moteur de graphe

Les messages d'erreur Neptune renvoient un code d'erreur HTTP pertinent et une réponse au format JSON.

```
HTTP/1.1 400 Bad Request
x-amzn-RequestId: LDM6CJP8RMQ1FHKSC1RBVJFPNVV4KQNSO5AEMF66Q9ASUAAJG
Content-Type: application/x-amz-json-1.0
Content-Length: 465
Date: Thu, 15 Mar 2017 23:56:23 GMT

{
    "requestId": "0dbcded3-a9a1-4a25-b419-828c46342e47",
    "code": "ReadOnlyViolationException",
    "detailedMessage": "The request is rejected because it violates some read-only
 restriction, such as a designation of a replica as read-only."
}
```

Erreurs de requête du moteur de graphe

Le tableau suivant contient le code d'erreur, le message et le statut HTTP.

Il indique également s'il convient de réessayer la demande. Généralement, il convient de réessayer la demande si elle peut aboutir lors d'une nouvelle tentative.

Code d'erreur du service Neptune	Statut HTTP	OK pour réessayer ?	Message
`AccessDeniedException`	403	Non	Échec de l'authentification ou de l'autorisation.
`BadRequestException`	400	Non	La demande n'a pas pu être exécutée.
`BadRequestException`	400	Non	La requête dépasse la valeur maximale autorisée de 157286400 octets.
`CancelledByUserException`	500	Oui	Le traitement de la demande a été annulé par un client autorisé.
`ConcurrentModificationException`	500	Oui	Le traitement de la demande a échoué en raison d'un conflit de modification. Le client doit relancer la demande.
`ConstraintViolationException`	400	Oui	Le moteur de requête a détecté, lors de l'exécution de la demande, que l'exécution d'une opération est impossible sans enfreindre certaines contraintes d'intégri

Code d'erreur du service Neptune	Statut HTTP	OK pour réessayer ?	Message
			té des données, telles que la persistance des sommets entrants et sortants lors de l'ajout d'un arc. Ces conditions sont généralement observées s'il y a des modifications simultanées du graphe, et sont transitoires. Le client doit relancer la demande.
`FailureByQueryException`	500	Oui	L'appel `fail()` a entraîné l'échec du traitement des demandes. Le client peut réessayer la demande en supposant que la logique de la requête devrait réussir lors d'une tentative ultérieure. Par conséquent, le choix de réessayer sera spécifique à la requête.
`InternalFailureException`	500	Non	Le traitement de la demande a échoué.

Code d'erreur du service Neptune	Statut HTTP	OK pour réessayer ?	Message
`InvalidNumericDataException`	400	Non	Utilisation non valide de données numériques qui ne peuvent pas être représentées dans une taille de stockage de 64 bits.
`InvalidParameterException`	400	Non	Une out-of-range valeur non valide a été fournie pour un paramètre d'entrée ou une syntaxe non valide dans un fichier RDF fourni.
`MalformedQueryException`	400	Non	La demande est rejetée, car elle contient une requête dont la syntaxe est incorrecte ou dont une validation supplémentaire échoue.
`MemoryLimitExceededException`	500	Oui	Le traitement de la demande a échoué en raison d'un manque de mémoire, mais pourra être réessayé lorsque le serveur sera moins occupé.

Code d'erreur du service Neptune	Statut HTTP	OK pour réessayer ?	Message
`MethodNotAllowedException`	405	Non	La demande est rejetée, car la méthode HTTP choisie n'est pas prise en charge par le point de terminaison utilisé.
`MissingParameterException`	400	Non	Un paramètre requis pour l'action spécifiée n'est pas fourni.
`QueryLimitExceededException`	500	Oui	Le traitement de la demande a échoué en raison du manque d'une ressource limitée, mais pourra être réessayé lorsque le serveur sera moins occupé.
`QueryLimitException`	400	Non	La taille de la requête dépasse la limite système.
`QueryTooLargeException`	400	Non	La demande a été rejetée, car son corps est trop volumineux.

Code d'erreur du service Neptune	Statut HTTP	OK pour réessayer ?	Message
`ReadOnlyViolationException`	400	Non	La demande est rejetée, car elle ne respecte pas une restriction en lecture seule, telle que la désignation d'un réplica en lecture seule.
`ThrottlingException`	500	Oui	La fréquence des demandes dépasse le débit maximum. OK pour réessayer.
`TimeLimitExceededException`	500	Peut-être	Le délai de traitement de la demande a expiré.
`TooManyRequestsException`	429	Oui	Le taux de demandes dépasse le débit maximal. OK pour réessayer.
`UnsupportedOperationException`	400	Non	La demande utilise une fonction ou une construction non prise en charge actuellement.

Important

Le fait de réessayer une erreur implique souvent des nuances importantes liées à la nature de l'échec, à la charge de travail et aux modèles d'erreur que vous observez. Pour obtenir

des conseils détaillés sur le développement d'une stratégie de réessai pratique, voirthe section called “Exceptions et nouvelles tentatives”.

Erreurs d'authentification IAM

Ces erreurs sont spécifiques à un cluster pour lequel l'authentification IAM est activée.

Le tableau suivant contient le code d'erreur, le message et le statut HTTP.

Code d'erreur du service Neptune	Statut HTTP	Message
Utilisateur/stratégie IAM incorrect	403	Vous ne disposez pas d'un accès suffisant pour effectuer cette action.
Région manquante ou incorrecte	403	Les informations d'identif ication doivent être limitées à une région valide, et non à « ». *region*
Nom de service manquant ou incorrect	403	Les informations d'identif ication doivent être définies sur le service correct : « neptune-db ».
En-tête hôte manquant ou incorrect/Signature non valide	403	La signature de demande que nous avons calculée ne correspond pas à la signature que vous avez fournie. Vérifiez votre clé d'accès AWS secrète et votre méthode de signature . Pour plus d'informations, consultez la documentation du service. L'en-tête hôte est manquant ou le nom d'hôte est incorrect.

Code d'erreur du service Neptune	Statut HTTP	Message
`X-Amz-Security-Token` manquant	403	«`x-amz-security-token` » est nommé en tant que `SignedHeader` , mais il n'existe pas dans la demande HTTP
En-tête d'autorisation manquant	403	La demande n'incluait pas l'en-tête d'autorisation obligatoire ou n'était pas correctement formée.
Jeton d'authentification manquant	403	Jeton d'authentification manquant.
Date passée	403	Signature expirée : *`20181011T213907Z`* elle est désormais antérieure à *`20181011T213915Z`* (*`20181011T214415Z -5 min`*.)
Date future	403	Signature non encore à jour : *`20500224T213559Z`* elle est encore postérieure à *`20181108T225925Z`* (*`20181108T225425Z +5 min`*.)
Format de date incorrect	403	La date doit être à un « format de base » ISO-8601. J'ai obtenu « *`date`* ». Consultez https://en.wikipedia.org/wiki/ISO_8601.

Code d'erreur du service Neptune	Statut HTTP	Message
Clé d'accès ou jeton de session inconnu/manquant	403	Le jeton de sécurité inclus dans la demande n'est pas valide.
Clé secrète inconnue/ manquante	403	La signature de demande que nous avons calculée ne correspond pas à la signature que vous avez fournie. Vérifiez votre clé d'accès AWS secrète et votre méthode de signature . Pour plus d'informations, consultez la documentation du service. L'en-tête hôte est manquant ou le nom d'hôte est incorrect.
`TooManyRequestsException`	429	Le taux de demandes dépasse le débit maximal. OK pour réessayer.

Messages et codes d'erreur liés à l'API de gestion de cluster de base de données

Ces erreurs Amazon Neptune sont associées à la création et à la APIs modification de ressources Neptune à l'aide du SDK et. AWS AWS CLI

Le tableau suivant contient le code d'erreur, le message et le statut HTTP.

Code d'erreur du service Neptune	Statut HTTP	Message
`AccessDeniedException`	403	Vous ne disposez pas d'un accès suffisant pour effectuer cette action.
`IncompleteSignature`	400	La signature de la demande n'est pas conforme aux AWS normes.
`InternalFailure`	500	Le traitement de la demande a échoué en raison d'une erreur inconnue, d'une exception ou d'un échec.
`InvalidAction`	400	L'action ou l'opération demandée n'est pas valide. Vérifiez que l'action est entrée correctement.
`InvalidClientTokenId`	403	Le certificat X.509 ou AWS l'ID de clé d'accès fourni n'existe pas dans nos archives.
`InvalidParameterCombination`	400	Des paramètres qui ne doivent pas être utilisés ensemble ont été utilisés conjointement.
`InvalidParameterValue`	400	Une valeur non valide ou une out-of-range valeur a été fournie pour le paramètre d'entrée.
`InvalidQueryParameter`	400	Une valeur non valide ou une out-of-range valeur a été fournie pour le paramètre d'entrée.

Code d'erreur du service Neptune	Statut HTTP	Message
`MalformedQueryString`	400	La chaîne de requête contient une erreur de syntaxe.
`MissingAction`	400	Il manque une action ou un paramètre requis dans la demande.
`MissingAuthenticationToken`	403	La demande doit contenir un identifiant de clé d' AWS accès valide (enregistré) ou un certificat X.509.
`MissingParameter`	400	Un paramètre requis pour l'action spécifiée n'est pas fourni.
`OptInRequired`	403	L'ID de clé d' AWS accès nécessite un abonnement au service.
`RequestExpired`	400	La demande est parvenue au service plus de 15 minutes après l'horodatage de la demande ou plus de 15 minutes après la date d'expiration de la demande (par exemple pour les demandes présignées URLs), ou l'horodatage de la demande est daté dans plus de 15 minutes dans le futur.
`ServiceUnavailable`	503	La requête a échoué en raison d'une défaillance temporaire du serveur.

Code d'erreur du service Neptune	Statut HTTP	Message
`ThrottlingException`	500	La demande a été refusée suite à une limitation des demandes.
`ValidationError`	400	L'entrée ne satisfait pas les contraintes spécifiées par un AWS service.

Message d'erreur et de flux liés au chargeur Neptune

Les messages suivants sont renvoyés par le point de terminaison `status` du chargeur Neptune. Pour de plus amples informations, veuillez consulter Get-Status API.

Le tableau suivant contient le code et la description du flux du chargeur.

Erreur ou code de flux	Description
LOAD_NOT_STARTED	Le chargement a été enregistré mais n'a pas démarré.
LOAD_IN_PROGRESS	Indique que le chargement est en cours et spécifie le nombre de fichiers actuellement chargés. Lorsque le chargeur analyse un fichier, il crée un ou plusieurs fragments à charger en parallèle. Étant donné qu'un seul fichier peut générer plusieurs fragments, le nombre de fichiers inclus dans ce message est généralement inférieur au nombre de threads utilisés par le processus de chargement en bloc.
LOAD_COMPLETED	Le chargement s'est terminé sans erreur ou avec des erreurs sous un seuil acceptable.

Erreur ou code de flux	Description
LOAD_CANCELLED_BY_USER	Le chargement a été annulé par l'utilisateur.
LOAD_CANCELLED_DUE_TO_ERRORS	Le chargement a été annulé par le système en raison d'erreurs.
LOAD_UNEXPECTED_ERROR	Le chargement a échoué avec une erreur inattendue.
LOAD_FAILED	Echec du chargement en raison d'une ou plusieurs erreurs.
LOAD_S3_READ_ERROR	Le flux a échoué en raison de problèmes de connectivité Amazon S3 intermittents ou transitoires. Si l'un des flux reçoit cette erreur, le statut de chargement global est défini sur LOAD_FAILED.
LOAD_S3_ACCESS_DENIED_ERROR	L'accès au compartiment S3 a été refusé. Si l'un des flux reçoit cette erreur, le statut de chargement global est défini sur LOAD_FAIL ED.
LOAD_COMMITTED_W_WRITE_CONFLICTS	Données chargées validées avec des conflits d'écriture non résolus. Le chargeur tente de résoudre les conflits d'écritures dans des transactions distinctes et met à jour le statut du flux à mesure que le chargement progresse. Si le statut de flux final est LOAD_COMMITTED_W_WRITE_CONF LICTS, essayez de reprendre le chargement qui réussira probablement sans conflit d'écritur e. Un conflit d'écriture n'est généralement pas associé à des données d'entrée incorrectes, mais des doublons dans les données peuvent augmenter la probabilité de conflits d'écritures.

Erreur ou code de flux	Description
LOAD_DATA_DEADLOCK	Le chargement a été annulé automatiquement en raison d'un blocage.
LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETED	Le flux a échoué, car le fichier a été supprimé ou mis à jour après le démarrage du chargement.
LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED	La demande de chargement n'a pas été exécutée car sa vérification de dépendance échoue.
LOAD_IN_QUEUE	La demande de chargement a été mise en file d'attente et attend d'être exécutée.
LOAD_FAILED_INVALID_REQUEST	Le chargement a échoué car la demande n'était pas valide (par exemple, le fichier spécifié n'existe source/bucket peut-être pas ou le format de fichier n'est pas valide).

Versions du moteur pour Amazon Neptune

Amazon Neptune publie régulièrement des mises à jour du moteur.

Pour déterminer la version du moteur qui est actuellement installée, vous pouvez utiliser l'API instance-status dans la console Neptune. Le numéro de version vous indique si vous exécutez une version majeure d'origine, une version mineure ou une version de correctif. Pour plus d'informations sur la numérotation des versions, consultez Numéros de version du moteur.

Pour plus d'informations sur les mises à jour en général, consultez Maintenance du cluster.

À partir de la version 1.3.0.0 du moteur, les versions du moteur auront la structure indiquée dans le tableau ci-dessous. Le numéro de version mineure est celui qui sera évalué pour le traitement `AutoMinorVersionUpgrade`.

Version	Versi de produ	Versi maje	Version mineure	Version de correctif	Statut	Libéré	Fin de vie	Mise à niveau vers :
1.4.7.0	1	4	7	0	actif	03/03/26/	03/06/2027/	N/A
1.4.6.3	1	4	6	3	actif	18/12/2025	03/06/2027/	1.4.7.0
1.4.6.2	1	4	6	2	actif	18/11/2025	03/06/2027/	1.4.7.0
1.4.6.1	1	4	6	1	actif	18/09/2025	06/03/2021	1.4.6.2
1.4.6.0	1	4	6	0	actif	02/09/2025	06/03/2021	1.4.6.1
1.4.5.1	1	4	5	1	actif	30/06/2025	06/03/2021	1.4.6.0
1.4.5.0	1	4	5	0	actif	09/04/2025-04	06/03/2021	1.4.5.1

Version	Versi de produ	Versi maje	Version mineure	Version de correctif	Statut	Libéré	Fin de vie	Mise à niveau vers :
1.4.4.0	1	4	4	0	actif	24/02/2025	06/03/2021	1.4.5.0
1.4.3.0	1	4	3	0	actif	21/01/2025	06/03/2021	1.4.4.0
1.4.2.0	1	4	2	0	actif	19-12-2024	06/03/2021	1.4.3.0
1.4.1.0	1	4	1	0	actif	21/11/2024	06/03/2021	1.4.2.0
1.4.0.0	1	4	0	0	actif	06/11/2024	06/03/2021	1.4.1.0
1.3.4.0	1	3	4	0	actif	2024-10-01	06/03/2021	1.4.0.0
1.3.3.0	1	3	3	0	actif	2024-08-05	06/03/2021	1.3.4.0
1.3.2.1	1	3	2	1	actif	20/06/2024	06/03/2021	1.3.3.0
1.3.2.0	1	3	2	0	actif	10/06/2022	06/03/2027/	1.3.2.1
1.3.1.0	1	3	1	0	actif	06/03/2024	06/03/2027/	1.3.2.1
1.3.0.0	1	3	0	0	actif	15/11/2023	06/03/2027/	1.3.2.1

Le tableau ci-dessous répertorie toutes les versions du moteur depuis la version 1.0.1.0, ainsi que des informations sur les versions end-of-life. Vous pouvez utiliser les dates suivantes pour planifier vos cycles de test et de mise à niveau.

Version	Version majeur	Version mineur	Statut	Libéré	Fin de vie	Mise à niveau vers :
1.2.1.2	1.2	1.2	actif	2024-08-05	30/06/2026/	1.3.0.0
1.2.1.1	1.2	1.1	actif	11/03/2024	30/06/2026/	1.3.0.0
1.2.1.0	1.2	1.0	actif	2023-03-08	30/06/2026/	1.3.0.0
1.2.0.2	1.2	0.2	actif	16 novembre	30/06/2026/	1.3.0.0
1.2.0.1	1.2	0.1	actif	26/10	30/06/2026/	1.3.0.0
1.2.0.0	1.2	0.0	actif	07-21	30/06/2026/	1.3.0.0
1.1.1.0	1.1	1.0	actif	19/04/2018	30/06/2026/	1.2.1.0
1.1.0.0	1.1	0.0	obsolète	2021-11-19	15/03/2025	1.1.1.0
1.0.5.1	1.0	5.1	obsolète	2021-10-01	30/01/2023	1.1.0.0
1.0.5.0	1.0	5.0	obsolète	2021-07-27	30/01/2023	1.1.0.0
1.0.4.2	1.0	4.2	obsolète	01-06-2021	30/01/2023	1.1.0.0
1.0.4.1	1.0	4.1	obsolète	2020-12-08	30/01/2023	1.1.0.0
1.0.4.0	1.0	4.0	obsolète	2020-10-12	30/01/2023	1.1.0.0
1.0.3.0	1.0	3.0	obsolète	2020-08-03	30/01/2023	1.1.0.0
1.0.2.2	1.0	2.2	obsolète	09 mars 2020	07-29	1.0.3.0
1.0.2.1	1.0	2.1	obsolète	22/11/2019	07-29	1.0.3.0
1.0.2.0	1.0	2.0	obsolète	08/11/2019	19/05/2020	1.0.3.0

Version	Version majeure	Version mineure	Statut	Libéré	Fin de vie	Mise à niveau vers :
1.0.1.2	1.0	1.2	obsolète	2019-10-15	—	—
1.0.1.1	1.0	1.1	obsolète	13/08/2019	—	—
1.0.1.0.*	1.0	1.0.*	obsolète	02/07/2021 et avant	—	—

end-of-lifePlanification des principales versions du moteur

Les versions du moteur Neptune atteignent presque toujours leur fin de vie à la fin d'un trimestre du calendrier civil, à l'exception des cas où des problèmes importants de sécurité ou de disponibilité surviennent.

Lorsqu'une version du moteur arrive en fin de vie, vous devez mettre à niveau votre base de données Neptune vers une version plus récente.

En général, les versions du moteur Neptune restent disponibles comme suit :

- Versions mineures du moteur : les versions mineures du moteur restent disponibles pendant au moins six mois après leur sortie.
- Versions majeures du moteur : les versions majeures du moteur restent disponibles pendant au moins 12 mois après leur sortie.

Au moins 3 mois avant la fin de vie d'une version du moteur, AWS vous enverrez une notification automatique par e-mail à l'adresse e-mail associée à votre AWS compte et publierez le même message sur votre AWS Health Dashboard. Cela vous laisse ainsi le temps de planifier et de préparer la mise à niveau.

Lorsqu'une version du moteur arrive en fin de vie, vous ne pouvez plus créer de clusters ni d'instances à l'aide de cette version. L'autoscaling ne peut plus créer d'instances à l'aide de cette version non plus.

Une version du moteur qui arrive à sa date de fin de vie effective est automatiquement mise à niveau pendant une fenêtre de maintenance. Le message qui vous est envoyé trois mois avant la fin de vie de la version du moteur contient des informations sur les implications de cette mise à jour

automatique, notamment sur la version de la mise à niveau qui aura lieu automatiquement, l'impact sur vos clusters de bases de données et les actions que nous recommandons.

> **Important**
>
> Vous êtes responsable de la mise à jour constante des versions de votre moteur de base de données. AWS invite tous les clients à mettre à niveau leurs bases de données vers la dernière version du moteur afin de bénéficier des garanties de sécurité, de confidentialité et de disponibilité les plus récentes. Si vous utilisez votre base de données sur un moteur ou un logiciel non pris en charge après la date d'obsolescence (ce que l'on considère comme un « ancien moteur »), vous êtes exposé à un risque accru de problèmes de sécurité, de confidentialité et d'exploitation, y compris des interruptions de service.
> L'exploitation de votre base de données sur n'importe quel moteur est soumise à l'accord régissant votre utilisation des AWS services. Les anciens moteurs ne sont généralement pas disponibles. AWS ne fournit plus de support à l'Legacy Engine et AWS peut imposer des limites à l'accès ou à l'utilisation de tout Legacy Engine à tout moment, s'il est AWS déterminé que l'Legacy Engine présente un risque de sécurité ou de responsabilité, ou un risque de préjudice AWS, pour les Services, ses filiales ou tout tiers. Votre décision de continuer à exécuter votre contenu dans un ancien moteur pourrait rendre votre contenu indisponible, corrompu ou irrécupérable. Les bases de données exécutées sur un ancien moteur sont soumises à des exceptions au contrat de niveau de service (SLA).
> LES BASES DE DONNÉES ET LES LOGICIELS ASSOCIÉS EXÉCUTÉS SUR UN ANCIEN MOTEUR CONTIENNENT DES BOGUES, DES ERREURS, DES DÉFAUTS ET DES COMPOSANTS AND/OR NUISIBLES. EN CONSÉQUENCE, ET NONOBSTANT TOUTE DISPOSITION CONTRAIRE DANS LE CONTRAT OU LES CONDITIONS DE SERVICE, AWS FOURNIT L'ANCIEN MOTEUR « TEL QUEL ».

Version 1.4.7.0 du moteur Amazon Neptune (03/03/2026/)

Depuis le 03/03/2020, la version 1.4.7.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Nouvelles fonctionnalités pour cette version du moteur

- OpenCypher est lu depuis le support S3 pour les fichiers Parquet et CSV via OC. Consultez la the section called “neptune.read ()” documentation.

- Fonctions de requête géospatiale OpenCypher. Cette version inclut 12 fonctions de types spatiaux basées sur la norme ISO/IEC 13249-3:2016, un nouveau type de propriété de géométrie pour POINT stocké dans un nouvel index géopétrique pour une récupération rapide, et la prise en charge du format texte (WKT). Well-Known Consultez le Données spatiales et la Fonctions spatiales documentation.

Améliorations de cette version du moteur

- Performances de requête améliorées pour les sous-requêtes SPARQL qui renvoient de petits ensembles de résultats, y compris ceux dont les valeurs LIMIT sont faibles
- Performances de requête améliorées dans les cas où les variables sont limitées par un très grand nombre de valeurs constantes (par exemple, par une clause SPARQL VALUES ou une clause OpenCypher UNWIND)
- Améliorations apportées aux requêtes d'insertion à faible latence grâce à certaines optimisations des insertions dans les dictionnaires
- De nouvelles étapes du langage G705 ont été ajoutées au moteur DFE (voir la couverture des étapes Gremlin dans DFE).
  - Trajectoire et étapes de traversée : `order(local)`
  - Étapes d'agrégation et de collecte : `dedup(local)`
- Amélioration des performances pour les OpenCypher requêtes, notamment`COLLECT(DISTINCT ...)`. La réécriture décrite dans Réécriture des requêtes COLLECT (DISTINCT...) n'est plus nécessaire lors de l'utilisation de la version 1.4.7.0 ou ultérieure du moteur.

Défauts corrigés dans cette version du moteur

Correctifs généraux :

- Correction du fait que le chargement en masse ne répondait pas lors du chargement d'un grand nombre de fichiers Edge
- Correction d'un problème global d'application de correctifs aux clusters de bases de données qui affectait les mises à jour des clusters secondaires à partir des versions 1.4.0.0, 1.4.1.0 et 1.4.2.0.

Query-Language Versions prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.4.7.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.4.7.0 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.7.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.7.0 ^
    --allow-major-version-upgrade ^
```

```
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le paramètre allow-major-version-upgrade est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.4.6.3 du moteur Amazon Neptune (18/12/2025)

Depuis le 18 décembre 2025, la version 1.4.6.3 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur (1.4.6.3)

Corrections générales

- Certaines exceptions classées par erreur comme des erreurs internes au serveur sont désormais correctement signalées comme des exceptions de mémoire insuffisante lors de l'exécution du moteur de flux de données (DFE).
- Correction d'un problème qui pouvait entraîner l'échec des instances de la base de données Neptune au démarrage lorsque la validation intégrée de l'identifiant Edge rencontrait des types d'entrées de dictionnaire inattendus. La validation intégrée de l'identifiant Edge ne s'exécute désormais que pour les configurations du moteur de requête Gremlin et gère de manière élégante les entrées du dictionnaire autres que les URI.
- Correction d'un problème qui empêchait le chargeur groupé Neptune de se connecter à S3 dans certaines régions en raison d'une résolution incorrecte des points de terminaison.

Correctifs apportés à openCypher

- Correction d'un crash moteur qui se produisait parfois lors de requêtes utilisant`CALL`.
- Délai d'expiration et gestion des annulations fixes pour les requêtes de mutation.
- Correction d'un bogue à cause duquel la `MERGE` clause donnait des résultats incorrects lorsque l'une des valeurs de propriété transmises `MERGE` avait une valeur nulle.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.4.6.3, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.4.6.3 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.6.3 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.6.3 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.4.6.2 du moteur Amazon Neptune (18/11/2025)

Depuis le 18 novembre 2025, la version 1.4.6.2 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

Corrections générales

- Certaines exceptions classées par erreur en tant qu'exceptions de défaillance interne (IFEs) sont désormais correctement signalées en tant qu'exceptions de modification simultanée (CMEs) lors de l'exécution du moteur de flux de données (DFE).

Correctifs apportés à Gremlin

- Stabilité du moteur améliorée pour Gremlin par rapport au DFE

Correctifs apportés à openCypher

- Correction d'un problème introduit dans la version 1.4.6.0 du moteur Neptune en raison duquel les requêtes OpenCypher MERGE géraient incorrectement les directions des relations entrantes (←). Les requêtes utilisant des modèles tels que MERGE (n) ← [:type] - (m) créent désormais des relations dans la bonne direction.
- Correction d'un échec d'une opération de validation sur les transactions du rédacteur qui n'étaient pas correctement annulées.
- Les plans de requêtes optimisés sont désormais pris en charge pour les requêtes MERGE qui font référence à des valeurs paramétrées.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.4.6.2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.4.6.2 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
```

```
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.6.2 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.6.2 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations,

l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.4.6.1 du moteur Amazon Neptune (18/09/2025)

Depuis le 18/09/2025, la version 1.4.6.1 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

Corrections générales

- Suppression des vérifications réseau pour les clusters utilisant des plages d'adresses IP privées non conformes à la RFC 1918 introduites dans la version. Version 1.4.6.0 du moteur Amazon Neptune (02/09/2020)

Correctifs apportés à Gremlin

- Correction d'un problème de gestion des connexions Websocket avec les transactions.
- Correction d'un rare problème de redémarrage d'instance lors de l'utilisation du mode DFE de gremlin

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.4.6.1, assurez-vous que votre projet est compatible avec les versions du langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.4.6.1 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.6.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.6.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.4.6.0 du moteur Amazon Neptune (02/09/2020)

Depuis le 02/09/2020, la version 1.4.6.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Warning

La version du moteur 1.4.6.0 inclut de nouvelles vérifications réseau pour les clusters qui utilisent des plages d'adresses IP privées non conformes à la RFC 1918 pour les VPC de base de données sans authentification IAM. Si vous avez cette configuration VPC et IAM, vous devrez mettre à jour votre base de données VPC pour utiliser les plages d'adresses IP privées RFC 1918 et and/or activer l'authentification IAM afin d'éviter les erreurs lors des requêtes après la mise à niveau vers la version 1.4.6.0.

Nouvelles fonctionnalités de cette version du moteur

- Connectez-vous à Neptune à l'aide de points de terminaison publics. Pour plus d'informations, consultez la section Points de terminaison publics de Neptune.

Améliorations apportées à cette version du moteur

Améliorations générales

- Performances SPARQL améliorées pour les opérations de mise à jour.
- OpenCypher Performances améliorées pour les opérations `CREATE`, `MERGE`, et `SET` (mutations).
- OpenCypher Performances améliorées pour les opérations de sous-requête CALL.

Améliorations d'openCypher

- Ajout d'un nouvel indice de requête pour prendre en charge le délai d'expiration au niveau de la requête.

Défauts corrigés dans cette version du moteur

Correctifs apportés à Gremlin

- Les connexions aux sessions G705 doivent se faire sur le même canal qui les a créées, ce qui signifie qu'il n'est pas possible de connecter plusieurs instances clientes à la même session.
- Les sessions G705 se sont toujours fermées lorsque le client ferme, mais elles se ferment désormais également en cas de fermeture de la connexion initiée par le serveur, ce qui empêche toute reconnexion involontaire ou inattendue.
- Correction de fuites de mémoire pour les requêtes G705 lisant des données de type blob volumineux.

Correctifs apportés à openCypher

- Correction de la gestion des variables après une `CALL` sous-requête.
- Correction d'un problème lié à `reduce` la fonction permettant de gérer correctement les scénarios de dépassement arithmétique.
- Correction d'une fuite de mémoire affectant les requêtes paramétrées lorsque le cache du plan de requête est activé.
- Correction d'un problème lié à l'`NOT EXISTS`utilisation dans les `WHERE` clauses complexes.
- Correction du fait que l'exception de mémoire simultanée (CMEs) était mal signalée comme. BadRequestException

Correctifs apportés à SPARQL

- Correction d'un message d'erreur pour SPARQL `LOAD/UNLOAD` lorsque la source distante n'est pas disponible.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.4.6.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.4.6.0 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.6.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.6.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.4.5.1 du moteur Amazon Neptune (30/06/2020)

Au 30 juin 2020, la version 1.4.5.1 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

Corrections générales

- Correction d'un problème lié au dimensionnement sans serveur qui avait un impact sur tous les langages de requête.
- Correction d'un problème en raison duquel une erreur interne était renvoyée sur certaines OpenCypher requêtes lors de l'exécution `collect(distinct())` d'une liste.
- Correction d'un problème en raison duquel une requête FTS simultanée entraînait l'arrêt d'autres requêtes FTS simultanées en cas d'utilisation OpenSearch sans serveur.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.4.5.1, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.4.5.1 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez

mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.5.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.5.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise

à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the instance before
>   proceeding with the upgrade.
> ```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.4.5.0 du moteur Amazon Neptune (2025-04-09)

Depuis le 2025-04-09, la version 1.4.5.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

⚠ Warning

Nous avons temporairement suspendu les mises à niveau vers la version 1.4.5.0 en raison de problèmes pouvant survenir dans certaines configurations sans serveur. Nous vous recommandons de passer à la version du moteur 1.4.5.1. Les mises à niveau vers la version 1.4.5.0 ont été temporairement désactivées.

Nouvelles fonctionnalités de cette version du moteur

- De nouvelles étapes du langage G705 ont été ajoutées au moteur DFE.
  - Chemin et étapes de traversée : AsDate (), DateAdd (), DateDiff (), fail (), Inject (), label (), chemin (), projet (), repeat (), sack (), select (), unfold (), disjonct (), drop (), identité (), intersection (), longueur (), boucles (), barrière (), ordre (), range (), reverse (), sample (), cap (), split (), filter (), flatMap (), map (), sideEffect (), union (), index ()
  - Étapes d'agrégation et de collecte : agrégation (globale), combinaison (), compte (), déduplication (globale), pliage (), groupe (), groupCount ()
  - Étapes mathématiques : max (), moyenne (), min (), somme ()
  - Étapes des éléments : otherV (), ElementMap (), élément (), V (), out (), in (), both (), outE (), iNE (), bothe (), outV (), inV (), bothV (), otherV ()

- Étapes relatives aux propriétés : propriétés (), clé (), valueMap (), valeur ()
- Étapes de filtrage : and (), coalesce (), coin (), is (), local (), none (), not () ou (), where ()
- Étapes de manipulation des chaînes : concat (), LTrim (), RTrim (), substring (), toLower (), toUpper (), trim ()
- Prédicats :
  - Comparez : eq, neq, lt, lte, gt, gte
  - Contient : dedans, sans
  - texTP : se terminant par, contenant,,, notStartingWith non contenant notEndingWith
  - P : et, ou, entre, à l'extérieur, à l'intérieur

Pour plus de détails sur toutes les étapes G705 disponibles dans DFE, reportez-vous à. Couverture des étapes de Garmlin en PDF

Améliorations apportées à cette version du moteur

Améliorations générales

- Amélioration de la lenteur du temps d'attente pour le verrouillage du journal des requêtes. Les journaux de requêtes lents incluent désormais des mesures de temps d'attente pour les verrous partagés et exclusifs. Ils sont stockés dans le cadre de chaque transaction en cas de promotion paresseuse en lecture-écriture. Ces métriques apparaissent dans la section StorageCounters des journaux des requêtes lentes.
- Suppression du support pour les suites de chiffrement suivantes :
  - TLS_ECDHE_RSA_WITH_AES_128_CBC_ SHA256
  - TLS_ECDHE_RSA_WITH_AES_256_CBC_ SHA384
  - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_ SHA256
  - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_ SHA384

Améliorations de Gremlin

- De nombreuses nouvelles étapes ont été ajoutées au langage Gremlin. Pour de plus amples informations, veuillez consulter Couverture des étapes de Garmlin en PDF.

Améliorations d'openCypher

- Améliorations des performances CREATE, MERGE et SET (mutations).
- Améliorations des performances des sous-requêtes CALL.
- Support de l'en-tête de suivi HTTP pour les réponses OpenCypher en plusieurs parties. Pour plus d'informations, consultez la section En-têtes de suivi HTTP facultatifs.
- Ajout de la fonction temporelle du jour, du mois et de l'année à OpenCypher. Pour plus d'informations, consultez la section Fonctions temporelles.

```
RETURN day(datetime('2021-06-03T01:48:14Z'))
{
  "results": [{
      "day(datetime('2021-06-03T01:48:14Z'))": 3
    }]
}
```

Défauts corrigés dans cette version du moteur

Corrections générales

- Correction d'un problème qui entraînait la Audit/SlowQueryLog suppression des fichiers journaux.

Correctifs apportés à Gremlin

- Correction d'un problème lié à l'exécution de requêtes G705 alors que la fonctionnalité Result Cache était désactivée. Une requête se terminant par iterate () renvoyait des résultats au lieu de renvoyer une réponse vide.
- Correction d'un problème lié au cache de résultats G705 causé par des requêtes simultanées avec la même clé. L'une des requêtes exécutées simultanément a renvoyé des résultats incorrects au lieu de renvoyer des résultats vides.
- Correction d'un problème lié aux requêtes d'exportation Amazon S3 qui entraînait l'échec d'une requête lors d'un téléchargement en plusieurs parties d'Amazon S3 en raison d'un délai d'expiration ou d'une annulation, en allongeant le temps de nettoyage.
- Correction d'un problème d'autorisation lié à l'exportation vers Amazon S3 par Gremlin.

Correctifs apportés à SPARQL

- Correction d'un problème dans le traitement des requêtes SPARQL qui déclaraient plusieurs bases, IRIs ce qui entraînait l'utilisation de la seule déclaration initiale.
- Correction d'un problème lié à la gestion des `REPLACE` fonctions SPARQL utilisant des chaînes de modèles non valides qui provoquaient le renvoi d'une erreur.
- Correction d'un problème lié à la gestion des `REPLACE` fonctions SPARQL à l'aide de l'indicateur case-insensitivity (`"i"`) avec les données Unicode.
- Correction d'un problème lié à l'analyse des requêtes SPARQL à l'aide de séquences d'échappement non valides `\u` et de `\U` points de code qui pouvait entraîner l'absence de réponse.
- Correction d'un problème dans la `IRI` fonction SPARQL qui ne résolvait pas toujours correctement la relation par IRIs rapport à l'IRI de base actuel.
- Correction d'un problème en raison duquel `SPARQL INSERT DATA` les `DELETE DATA` mises à jour utilisant des noms préfixés ne parvenaient pas à résoudre correctement la relation par IRIs rapport à l'IRI de base actuel.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.4.5.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.4.5.0 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez

mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.5.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.5.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise

à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the instance before
>   proceeding with the upgrade.
> ```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.4.4.0 du moteur Amazon Neptune (24/02/2025)

Au 24/02/2025, la version 1.4.4.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

⚠ Warning

Le cache du plan de requêtes n'est temporairement pas pris en charge pour l'exécution de requêtes paramétrées impliquant des valeurs de paramètres numériques, en raison d'un bogue dans la gestion des utilisations dupliquées d'un paramètre de type numérique dans la requête. Par exemple :

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
      UNION
      MATCH (n:show) WHERE n.duration>=$minutes RETURN n

      parameters={"minutes":130}
```

Les requêtes qui effectuent de nombreuses recherches d'index sur des déclarations ou des indices de dictionnaire peuvent entraîner une régression des performances de 5 %. Par exemple, le fait de compter tous les sommets ou d'obtenir le nombre `id` de tous les sommets ne serait pas affecté. L'obtention de toutes les propriétés de tous les sommets peut entraîner une régression allant jusqu'à 5 %.

Défauts corrigés dans cette version du moteur

Corrections générales

- Correction d'un problème en raison duquel la `WITH` clause contenant un astérisque (`*`) et des expressions d'alias provoquait une analyse incorrecte des requêtes dans OpenCypher.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.4.4.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.4.4.0 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.4.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.4.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.4.3.0 du moteur Amazon Neptune (21/01/2021)

Depuis le 21/01/2021, la version 1.4.3.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Warning

Le cache du plan de requêtes n'est temporairement pas pris en charge pour l'exécution de requêtes paramétrées impliquant des valeurs de paramètres numériques, en raison d'un bogue dans la gestion des utilisations dupliquées d'un paramètre de type numérique dans la requête. Par exemple :

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
      UNION
      MATCH (n:show) WHERE n.duration>=$minutes RETURN n

      parameters={"minutes":130}
```

Les requêtes qui effectuent de nombreuses recherches d'index sur des déclarations ou des indices de dictionnaire peuvent entraîner une régression des performances de 5 %. Par exemple, le fait de compter tous les sommets ou d'obtenir le nombre `id` de tous les sommets ne serait pas affecté. L'obtention de toutes les propriétés de tous les sommets peut entraîner une régression allant jusqu'à 5 %.

Nouvelles fonctionnalités de cette version du moteur

- Exportation des résultats de requêtes Gremlin vers Amazon S3. Exportation des résultats des requêtes G705 directement vers Amazon S3. Cette fonctionnalité vous permet de gérer efficacement les résultats de requêtes volumineux en les exportant vers un compartiment Amazon S3, au lieu de les renvoyer sous forme de réponse à une requête.

```
g.V().
    hasLabel('Comment').
    valueMap().
    call('neptune.query.exportToS3', [
    'destination': 's3://your-bucket/path/result.json',
    'format': 'GraphSONv3',
    'keyArn': 'optional-kms-key-arn'
```

```
])
```

- Instances R7i. La famille d'instances R7i, d'une taille maximale de 48 fois, est désormais disponible dans les régions suivantes :
  - ap-northeast-1 - Asie-Pacifique (Tokyo)
  - ap-northeast-2 - Asie-Pacifique (Séoul)
  - ap-south-1 - Asie-Pacifique (Mumbai)
  - ap-southeast-1 - Asie-Pacifique (Singapour)
  - ap-southeast-2 - Asie-Pacifique (Sydney)
  - ap-southeast-3 - Asie-Pacifique (Jakarta)
  - ca-central-1 - Canada (Centre)
  - eu-central-1 - Europe (Francfort)
  - eu-north-1 - Europe (Stockholm)
  - eu-south-2 - Europe (Espagne)
  - eu-west-1 - Europe (Irlande)
  - eu-west-2 - Europe (Londres)
  - eu-west-3 - Europe (Paris)
  - us-east-1 - USA Est (Virginie du Nord)
  - us-east-2 - Est des États-Unis (Ohio)
  - us-west-1 - USA Ouest (Californie du Nord)
  - us-west-2 - Ouest des États-Unis (Oregon)

Améliorations apportées à cette version du moteur

Améliorations générales

- Support en mode laboratoire pour la collecte des déchets dans les dictionnaires (GC).

  Lorsque cette option est activée, les entrées de dictionnaire inutilisées sont nettoyées par une tâche en arrière-plan. Il ne réduit pas`VolumeBytesUsed`, il libère de l'espace dans l'index pour de nouvelles insertions. Le taux de croissance `VolumeBytesUsed` est susceptible d'être inférieur lorsque le dictionnaire GC est activé par rapport à lorsqu'il ne l'est pas. Cela fonctionne pour les données du graphe de propriétés (insérées via Gremlin ou OpenCypher) lorsque le

`neptune_streams` paramètre n'est pas activé. Pour de plus amples informations, consultez Collecte des déchets du dictionnaire Neptune.

Défauts corrigés dans cette version du moteur

Corrections générales

- Correction de deux problèmes de fuite de mémoire affectant FreeableMemory l'utilisation du moteur DFE.

Correctifs apportés à openCypher

- Résolvez le problème avec MERGE ON MATCH/ON CREATE pour les lignes dupliquées.

```
UNWIND [1, 1] AS id
MERGE (n:Person {id: id})
  ON CREATE SET n.p = 5
  ON MATCH SET n.p = 6
```

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.4.3.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.4.3.0 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.3.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.3.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.4.2.0 du moteur Amazon Neptune (1912-2024)

Depuis le 19 décembre 2024-12, la version 1.4.2.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

⚠ Warning

Le cache du plan de requêtes n'est temporairement pas pris en charge pour l'exécution de requêtes paramétrées impliquant des valeurs de paramètres numériques, en raison d'un bogue dans la gestion des utilisations dupliquées d'un paramètre de type numérique dans la requête. Par exemple :

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
      UNION
      MATCH (n:show) WHERE n.duration>=$minutes RETURN n

      parameters={"minutes":130}
```

Les requêtes qui effectuent de nombreuses recherches d'index sur des déclarations ou des indices de dictionnaire peuvent entraîner une régression des performances de 5 %. Par exemple, le fait de compter tous les sommets ou d'obtenir le nombre `id` de tous les sommets ne serait pas affecté. L'obtention de toutes les propriétés de tous les sommets peut entraîner une régression allant jusqu'à 5 %.

Défauts corrigés dans cette version du moteur

Corrections générales

- Plan d'exécution et performances fixes pour les requêtes avec accès à plusieurs propriétés sur la même variable dans OPTIONAL MATCH et compréhension de liste. Par exemple :

```
MATCH (n)
  WHERE n.name = 'A'
OPTIONAL MATCH (n)-[:knows]->(m)
  WHERE m.name = 'C' or m.city = 'B'
RETURN m
```

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.4.2.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.4.2.0 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.2.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.2.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations,

l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.4.1.0 du moteur Amazon Neptune (21/11/2021)

Depuis le 21/11/2021, la version 1.4.1.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Warning**
>
> Le cache du plan de requêtes n'est temporairement pas pris en charge pour l'exécution de requêtes paramétrées impliquant des valeurs de paramètres numériques, en raison d'un bogue dans la gestion des utilisations dupliquées d'un paramètre de type numérique dans la requête. Par exemple :
>
> ```
> MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
>       UNION
>       MATCH (n:show) WHERE n.duration>=$minutes RETURN n
>
>       parameters={"minutes":130}
> ```
>
> Les requêtes qui effectuent de nombreuses recherches d'index sur des déclarations ou des indices de dictionnaire peuvent entraîner une régression des performances de 5 %. Par exemple, le fait de compter tous les sommets ou d'obtenir le nombre `id` de tous les sommets ne serait pas affecté. L'obtention de toutes les propriétés de tous les sommets peut entraîner une régression allant jusqu'à 5 %.

Nouvelles fonctionnalités de cette version du moteur

- Ajout de la prise en charge des `CALL` sous-requêtes avec une sous-requête en lecture seule, permettant l'exécution d'opérations dans un périmètre défini. Une `CALL` sous-requête est exécutée une fois pour chaque ligne entrante et les variables renvoyées dans une sous-requête sont disponibles dans toute la portée de la requête englobante. Les variables d'une portée externe peuvent être importées dans une `CALL` sous-requête à l'aide d'une `WITH` clause d'importation. Pour plus d'informations, consultez la section Prise en charge des sous-requêtes CALL dans Neptune.

```
MATCH (origin:airport {code:"AUS"})-[:route]->(stopover)
```

```
CALL {
  WITH stopover
  MATCH (stopover)-[r:route]->(destination)
  RETURN destination
  ORDER BY r.dist DESC LIMIT 2
}
RETURN stopover, destination
```

- Fonctions OpenCypher ajoutées. Nous introduisons huit nouvelles fonctions pour faciliter les chaînes, les opérations de collecte et le tri des collections. Il s'agit notamment de : `textIndexOf` `collToSetcollSubtract`,`collIntersection`,`collSort`,`collSortMaps`,,`collSortMulti`, et`collSortNodes`. Voir les fonctions Neptune OpenCypher pour la description, les paramètres d'entrée, la sortie et des exemples.

Améliorations apportées à cette version du moteur

Améliorations de Gremlin

- Nouveau paramètre du mode laboratoire`AccurateQRCMemoryEstimation`. Lorsqu'il est activé, le cache des résultats des requêtes G705 permet de mettre en cache les résultats des requêtes dans la base de données. Par défaut, une estimation approximative est utilisée pour déterminer la taille du résultat mis en cache. Lorsque ce paramètre en mode laboratoire `AccurateQRCMemoryEstimation` est activé, l'estimation de la taille des résultats mis en cache utilisera une estimation de taille précise au lieu d'une estimation approximative.
- Correction d'un problème lié à l'optimisation « non » du filtre dans les requêtes G705 exécutées sur le moteur d'exécution par défaut. Ce problème affectait les requêtes lorsque les contours étaient filtrés à l'aide de l'étape not () combinée à l'une des étapes OutV () /inv () /otherV (). Les exemples de requêtes incluent :
  - `g.E().hasLabel("knows").not(outV().hasId("5"))`
  - `g.V().has('airport','code','SDF').outE().where(not(otherV().has(id, within('1','5','7')))).count()`

Améliorations d'openCypher

- Performances améliorées pour les requêtes utilisant de grandes listes statiques ou des cartes. Certaines requêtes effectuées avec UNWIND sur une longue liste de cartes imbriquées utilisées

pour insérer/déplacer un nœud doté de propriétés permettent d'améliorer considérablement les performances.

- Introduit un nouvel indice de requête OpenCypher pour indiquer au moteur de supposer que les types de données sont cohérents pour les valeurs utilisées dans la requête. Consultez AssumeConsistentDataTypespour plus de détails sur le nouvel indice de requête OpenCypher.
- Introduit un ensemble de nouvelles fonctions OpenCypher pour gérer le texte et les valeurs de collection.

Défauts corrigés dans cette version du moteur

Correctifs apportés à Gremlin

- Correction d'un problème dans le chemin du code TinkerPop OSS qui permettait de créer une représentation en bytecode d'une requête de traversée lorsque l'une des `withStrategies()/withoutStrategies()/with()` étapes était utilisée sur un objet `GraphTraversalSource` « g ». Le problème a incorrectement ajouté de nouvelles instructions au Bytecode au lieu de remplacer les instructions existantes pour la même stratégie et a provoqué une incompatibilité des clés de cache lors de l'invalidation du cache de résultats pour effacer les résultats stockés.

Correctifs apportés à openCypher

- Comportement corrigé pour `` `~id`match `` les CREATE/MERGE/MATCH clauses in. Lorsque vous utilisez une `` `~id` `` valeur non valide, telle que des types nuls ou autres que des chaînes, une exception correcte est désormais émise pour les CREATE/MERGE clauses et aucun résultat n'est renvoyé pour une `MATCH` clause.
- IFE fixe lorsque l'utilisateur utilise une valeur d'un type non pris en charge avec des fonctions d'agrégation (par exemple sum (<string>)).
- Correction d'un problème en raison duquel certaines requêtes de mutation à faible latence provenant d'une charge de travail importante échouaient avec une OutOfMemory erreur.

Correctifs apportés à SPARQL

- Correction d'un problème de journal d'audit lors de la gestion des requêtes SPARQL contenant le `'%'` caractère.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.4.1.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.4.1.0 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.4.0.0 du moteur Amazon Neptune (06/11/2021)

Depuis le 06/11/2021, la version 1.4.0.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> La version 1.3.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version du moteur antérieure à la version 1.3.0.0 ou supérieure, vous devez recréer tous vos groupes de paramètres personnalisés et

groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres. `neptune1.3` Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1` ou `neptune1.2`, lesquels ne sont pas compatibles avec les versions 1.3.0.0 et supérieures. De même, vous devez utiliser des groupes de paramètres de cluster 1.4.0.0 pour les versions de moteur 1.4.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

Warning

Le cache du plan de requêtes n'est temporairement pas pris en charge pour l'exécution de requêtes paramétrées impliquant des valeurs de paramètres numériques, en raison d'un bogue dans la gestion des utilisations dupliquées d'un paramètre de type numérique dans la requête. Par exemple :

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
      UNION
      MATCH (n:show) WHERE n.duration>=$minutes RETURN n

      parameters={"minutes":130}
```

Les requêtes qui effectuent de nombreuses recherches d'index sur des déclarations ou des indices de dictionnaire peuvent entraîner une régression des performances de 5 %. Par exemple, le fait de compter tous les sommets ou d'obtenir le nombre `id` de tous les sommets ne serait pas affecté. L'obtention de toutes les propriétés de tous les sommets peut entraîner une régression allant jusqu'à 5 %.

Nouvelles fonctionnalités de cette version du moteur

- Lorsqu'une arête est ajoutée à un graphe de propriétés sans identifiant explicite, le serveur attribue par défaut un identifiant de bord basé sur l'UUID, qui est stocké dans le dictionnaire. Désormais, en définissant un nouveau paramètre de cluster`neptune_enable_server_generated_edge_id = 1`, le serveur attribuera IDs en utilisant un entier de 8 octets géré en interne, sans aucune surcharge du dictionnaire. Cela permet de réaliser des économies de stockage et d'améliorer les performances des requêtes sans aucune modification des requêtes. Cette fonctionnalité n'est actuellement prise en charge que pour les insertions via le langage de requête Gremlin.

- Ajout de la prise en charge de l'exécution des étapes Gremlin limit () dans les traversées imbriquées pour le moteur DFE.

```
g.V().project("foo").by(out().order().by(T.id).limit(1))
```

Améliorations apportées à cette version du moteur

Améliorations générales

- Neptune récupérera automatiquement l'espace d'annulation détenu par les transactions importantes une fois que la transaction sera terminée et que les journaux ne seront plus nécessaires pour la restauration.
- Support pour les répliques survivables de bases de données globales. Cette fonctionnalité permet au cluster secondaire de continuer à traiter les demandes de lecture lors du redémarrage d'une instance d'écriture sur le cluster principal. Auparavant, lorsqu'une instance d'enregistreur redémarrait, toutes les instances de lecteur d'un cluster secondaire redémarraient également. Avec cette version, les instances de lecteur de cluster secondaires continuent de traiter les demandes de lecture lors du redémarrage d'une instance d'écriture, ce qui améliore la disponibilité de lecture dans le cluster.
- Les journaux d'audit sont désormais écrits de manière synchrone, ce qui garantit que chaque requête est enregistrée. Cela peut affecter les performances pour les requêtes particulièrement volumineuses (> 100 Ko) ou les charges de travail à haut débit (> 1 000 qps).

Améliorations de Gremlin

- Le délai d'expiration par requête est imposé par défaut pour être inférieur au délai d'expiration au niveau du cluster. Dans une version précédente, cette vérification avait été introduite mais devait être explicitement activée via le paramètre lab-mode « ». StrictTimeoutValidation Dans cette version, « StrictTimeoutValidation » sera activé par défaut et devra être désactivé explicitement pour conserver l'ancien comportement.

Améliorations d'openCypher

- Dans une version précédente, nous avons introduit la prise en charge étendue du format datetime, activée via un paramètre `DatetimeMillisecond` du mode laboratoire. Cette prise en charge étendue du format date/heure est désormais activée par défaut.

Améliorations de SPARQL

- Nouvelles actions IAM explicites pour les autorisations de requête.

```
Previously:
COPY: WriteDataViaQuery & ReadDataViaQuery
MOVE: WriteDataViaQuery & DeleteDataViaQuery
DELETEINSERT: ReadDataViaQuery & DeleteDataViaQuery

 Now,
COPY: WriteDataViaQuery & ReadDataViaQuery & DeleteDataViaQuery
MOVE: WriteDataViaQuery & ReadDataViaQuery & DeleteDataViaQuery
DELETEINSERT: ReadDataViaQuery, WriteDataViaQuery if there is INSERT clause,
 DeleteDataViaQuery if there is DELETE clause.
```

Défauts corrigés dans cette version du moteur

Corrections générales

- Correction d'un problème lié aux instances sans serveur qui pouvait entraîner le redémarrage de la base de données lors de la mise à l'échelle.
- Correction d'un problème lié à la gestion des fichiers journaux d'audit qui pouvait rendre les fichiers journaux inaccessibles pour le téléchargement ou la rotation et, dans certains cas, augmenter l'utilisation du processeur.
- Correction d'un problème de requête lié à l'optimisation qui retardait la génération de la sortie cartographique dans le moteur DFE.
- Correction d'un problème qui provoquait des horodatages différents entre les journaux d'audit et la lenteur des journaux de requêtes.

Correctifs apportés à Gremlin

- Résolution d'un problème lié à la gestion des WebSocket connexions Gremlin, à savoir que les requêtes exécutées pendant une durée supérieure au délai d'inactivité de la connexion étaient interrompues prématurément. Cela a spécifiquement affecté les clients Python G705 utilisant le transport AIOHTTP.

Correctifs apportés à openCypher

- Correction d'un problème dans l'étape de collecte qui provoquait une exception de défaillance interne lorsque des valeurs nulles étaient présentes lors de la construction de la requête collect (distinct (n)).
- Correction d'un problème `NullPointerException` qui pouvait apparaître dans les requêtes lorsque le cache du plan de requêtes était activé.
- Correction d'un problème qui évaluait plus de données que nécessaire lorsqu'une requête contenait une clause LIMIT.
- Correction d'un problème en raison duquel l'utilisation d'opérations de plage (**<**, **<=**, **>**, **>=**) dans une requête paramétrée avec un cache de plan de requête produisait des résultats dupliqués.
- Correction d'un problème qui transposait les colonnes de résultats lorsque les opérations UNION et UNION ALL étaient effectuées à l'aide de connexions par boulon.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.4.0.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.4.0.0 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.3.4.0 du moteur Amazon Neptune (2024-10-01)

Depuis le 2024-10-01, la version 1.3.4.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Note

La version 1.3.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version du moteur antérieure à la version 1.3.0.0 ou supérieure, vous devez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres. `neptune1.3` Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1` ou `neptune1.2`, lesquels ne sont pas compatibles avec les versions 1.3.0.0 et supérieures. De même, vous devez utiliser des groupes de paramètres de cluster 1.4.0.0 pour les versions de moteur 1.4.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

Warning

Un problème a été détecté concernant les opérations de mise à jour de SPARQL 1.1 qui peuvent être présentes dans certaines conditions lorsque des opérateurs de mise à jour sont utilisés avec des politiques d'autorisation basées sur des actions. Si vous utilisez les opérations de mise à jour de SPARQL 1.1 avec des politiques d'autorisation basées sur des actions, nous vous recommandons de passer à la dernière version mineure du moteur Neptune (au moins 1.3.4.0) qui inclut un correctif pour ce problème.

Le cache du plan de requêtes a été temporairement désactivé pour les requêtes paramétrées impliquant des valeurs de paramètres numériques en raison d'un problème lié à la gestion des utilisations dupliquées d'un paramètre de type numérique, comme dans la requête suivante :

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$minutes RETURN n

parameters={"minutes":130}
```

Améliorations apportées à cette version du moteur

- Ajout de la prise en charge de l'exécution des étapes Gremlin limit () dans les traversées imbriquées pour le moteur DFE.
- Ajout de CloudWatch métriques liées au cache de résultats G705, comme indiqué ci-dessous, qui peuvent être utiles pour diagnostiquer et régler la latence du cache de résultats. Consultez Neptune Metrics pour plus de détails.

```
NumResultCacheHit
NumResultCacheMiss
ResultCacheSizeInBytes
ResultCacheItemCount
ResultCacheOldestItemTimestamp
ResultCacheNewestItemTimestamp
```

Défauts corrigés dans cette version du moteur

Améliorations générales

- Correction d'un problème à cause duquel, dans de rares cas, le moteur se bloquait au lieu de renvoyer une erreur de requête.

Correctifs apportés à Gremlin

- Nous avons amélioré le traitement des demandes et le signalement des erreurs lorsqu'un client ou un proxy envoie une demande de mise à niveau du Websocket via une connexion established/used HTTP (auparavant, 400 réponses contenant l'erreur « aucun script g705 fourni, code MissingParameterException » étaient renvoyées).
- Optimisation de la gestion des `mergeV` étapes grâce à des mises à jour de la valeur des propriétés de cardinalité uniques. Par exemple, la requête ci-dessous est désormais prise en charge de manière native dans Neptune.

```
g.mergeV([(T.id): 1234]). option(onMatch, ['age': single(20), 'name':
 single('alice'), 'city': set('miami')])
```

- Correction d'un problème d'évaluation des requêtes G705 DFE qui provoquait l'échec des requêtes avec un. `InternalFailureException` Cette erreur s'est produite avec certains modèles de`select`, comme indiqué dans l'exemple suivant :

```
g.V("1").as("a").as("b").select("a","b").dedup()
```

Correctifs apportés à openCypher

- Correction d'un problème en raison duquel l'exécution `collect(distinct())` en présence de valeurs nulles provoquait le renvoi d'une erreur.
- Problème résolu : l'exécution d'une requête paramétrée contenant un filtre de plage (**</<=/>/>=** par rapport à la valeur du paramètre) produisait des résultats. duplicate/missing
- Correction d'un bogue en raison duquel le moteur DFE produisait plus de résultats que ce qui était demandé dans les requêtes limites, ce qui pouvait entraîner des erreurs de mémoire insuffisante.

Correctifs apportés à SPARQL

- Correction d'un problème en raison duquel l'exécution d'une requête de mise à jour SPARQL fédérée sur des clusters dotés de l'authentification IAM entraînait le renvoi d'une erreur.
- Autorisations basées sur des actions fixes pour les opérations de mise à jour de SPARQL 1.1.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.3.4.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.3.4.0 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.4.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.4.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.3.3.0 du moteur Amazon Neptune (05/08/2021)

Depuis le 5 août 2021, la version 1.3.3.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> La version 1.3.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version du moteur antérieure à la version 1.3.0.0

ou supérieure, vous devez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres. `neptune1.3` Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1` ou `neptune1.2`, lesquels ne sont pas compatibles avec les versions 1.3.0.0 et supérieures. De même, vous devez utiliser des groupes de paramètres de cluster 1.4.0.0 pour les versions de moteur 1.4.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

⚠ Warning

La version 1.3.3.0 du moteur a introduit des problèmes potentiels dont vous devez être conscient. Consultez la section ci-dessous Atténuation des problèmes dans la version 1.3.3.0 pour plus d'informations.

Défauts corrigés dans cette version du moteur

Améliorations générales

- Correction d'un problème d'instabilité du moteur lorsque le cache de prédicats contient un grand nombre de prédicats.

Correctifs apportés à openCypher

- Correction d'un problème qui empêchait l'exécution d'une requête après le déclenchement d'une exception interne.
- Correction d'un problème en raison duquel une requête pouvait échouer avec une exception interne lors de l'utilisation du cache du plan de requête.

Correctifs apportés à SPARQL

- Correction d'un problème lié au protocole HTTP SPARQL 1.1 Graph Store (GSP) qui pouvait être présent dans certaines conditions lorsque le GSP est utilisé avec des politiques d'autorisation basées sur des actions.

Atténuation des problèmes dans la version 1.3.3.0

- Les requêtes utilisant des valeurs de filtre numériques peuvent renvoyer des résultats incorrects lors de l'utilisation du cache du plan de requêtes. Pour éviter ce problème, utilisez l'indice de requête `QUERY:PLANCACHE "disabled"` pour ignorer le cache du plan de requête. Par exemple, utilisez :

```
USING QUERY:PLANCACHE "disabled"
MATCH (n:person)
WHERE n.yearOfBirth > $year
RETURN n

parameters={"year":1950}
```

- Les requêtes utilisant le même nom de paramètre plusieurs fois peuvent échouer avec l'erreur`Parameter name should not be a number and/or contain _internal_ or _modified_user_ string within it. These are reserved for planCache. Otherwise, rerun with HTTP parameter planCache=disabled`. Dans ce cas, ignorez le cache du plan de requête comme ci-dessus ou dupliquez les paramètres comme dans cet exemple :

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$minutes RETURN n

parameters={"minutes":130}
```

  Utilisez le conseil `QUERY:PLANCACHE "disabled"` ou modifiez les paramètres :

```
MATCH (n:movie) WHERE n.runtime>=$rt_min RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$dur_min RETURN n

parameters={"rt_min":130, "dur_min":130}
```

- Les requêtes exécutées avec le protocole Bolt peuvent produire des résultats incorrects s'il s'agit d'une requête UNION ou UNION ALL. Pour éviter ce problème, envisagez d'exécuter la requête en question avec le point de terminaison HTTP. Vous pouvez également exécuter chaque partie de l'union séparément lorsque vous utilisez le protocole Bolt.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.3.3.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.3.3.0 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.3.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.3.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.3.2.1 du moteur Amazon Neptune (20/06/2021)

Depuis le 2024-06-20, la version 1.3.2.1 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> La version 1.3.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version du moteur antérieure à la version 1.3.0.0

ou supérieure, vous devez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres. `neptune1.3` Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1` ou `neptune1.2`, lesquels ne sont pas compatibles avec les versions 1.3.0.0 et supérieures. De même, vous devez utiliser des groupes de paramètres de cluster 1.4.0.0 pour les versions de moteur 1.4.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

Warning

La version 1.3.2.1 du moteur a introduit des problèmes potentiels dont vous devez être conscient. Consultez la section ci-dessous Atténuation des problèmes dans la version 1.3.2.1 pour plus d'informations.

Défauts corrigés dans cette version du moteur

Correctifs apportés à openCypher

- Un bogue a été détecté dans la fonctionnalité de cache du plan de requêtes pour les requêtes paramétrées contenant une `WITH` clause interne ayant `SKIP` et en `LIMIT` tant que paramètres. Les SKIP/LIMIT valeurs n'étaient pas correctement paramétrées et, par conséquent, les exécutions ultérieures du même plan de requête mis en cache avec des valeurs de paramètres différentes renverraient toujours les mêmes résultats que lors de la première exécution. Cela a été corrigé.

```
# insert some nodes
UNWIND range(1, 10) as i CREATE (s {name: i}) RETURN s

# sample query
MATCH (p)
WITH p ORDER BY p.name SKIP $s LIMIT $l
RETURN p.name as res

# first time executing with {"s": 2, "l": 1}
{
  "results" : [ {
    "res" : 3
  } ]
```

```
}

# second time executing with {"s": 2, "l": 10}
# due to bug, produces
{
  "results" : [ {
    "res" : 3
  } ]
}
# with fix, produces correct results:
{
  "results" : [ {
    "res" : 3
  }, {
    "res" : 4
  }, {
    "res" : 5
  }, {
    "res" : 6
  }, {
    "res" : 7
  }, {
    "res" : 8
  }, {
    "res" : 9
  }, {
    "res" : 10
  } ]
}%
```

- Correction d'un bogue où les requêtes de mutation paramétrées lançaient un `InternalFailureException` message lorsque le paramètre passé n'était pas déjà présent dans la base de données.
- Correction d'un bug à cause duquel les requêtes Bolt paramétrées restaient bloquées après avoir atteint une condition de course lors du nettoyage des ressources de requête.

Les modifications apportées au 1.3.2.1 sont reportées du 1.3.2.0

Améliorations apportées depuis la version 1.3.2.0 du moteur

Améliorations générales

- Support de la version 1.3 de TLS, y compris les suites de chiffrement TLS_AES_128_GCM_ et TLS_AES_256_GCM_SHA256 . SHA384 Le protocole TLS 1.3 est une option, le protocole TLS 1.2 reste le minimum.
- Le support étendu d'OpenCypher pour le format datetime est disponible en mode lab_pour cette version. Nous vous encourageons à le tester.

Améliorations apportées à Gremlin

- TinkerPop Mise à niveau 3.7.x
  - Fournit une extension importante du langage Gremlin.
    - Nouvelles étapes pour le traitement des chaînes, des listes et des dates.
    - Nouvelle syntaxe pour spécifier la cardinalité au sein de l'`mergeV()`étape.
    - `union()`peut désormais être utilisé comme étape de départ.
    - Pour en savoir plus sur les modifications apportées à la version 3.7.x, consultez la documentation de TinkerPop mise à niveau.
  - Lors de la mise à niveau des pilotes de langage Gremlin du client pour Java, notez que les classes du sérialiseur n'ont pas été renommées. Vous devrez mettre à jour le nom des packages et des classes dans vos fichiers de configuration et dans le code, si cela est spécifié.
- `StrictTimeoutValidation`(uniquement lorsqu'il est activé via labmode `StrictTimeoutValidation` en incluant`StrictTimeoutValidation=enabled`) : Lorsque le `StrictTimeoutValidation` paramètre a une valeur de`enabled`, une valeur de délai par requête spécifiée en tant qu'option de demande ou indice de requête ne peut pas dépasser la valeur définie globalement dans le groupe de paramètres. Dans ce cas, Neptune lancera un. `InvalidParameterException` Ce paramètre peut être confirmé dans une réponse sur le `/status` terminal lorsque la valeur est`disabled`, et dans les versions 1.3.2.0 et 1.3.2.1 de Neptune, la valeur par défaut de ce paramètre est. `Disabled`

Améliorations d'openCypher

- La version 1.3.2.0 du moteur Amazon Neptune fournit un débit jusqu'à 9 fois plus rapide et 10 fois plus élevé pour les performances des requêtes OpenCypher par rapport aux versions précédentes du moteur.
- Requêtes à faible latence et amélioration des performances de débit : améliorations des performances globales pour les requêtes OpenCypher à faible latence. La nouvelle version améliore également le débit de ces requêtes. Les améliorations sont plus importantes lorsque des requêtes paramétrées sont utilisées.
- Support pour le cache du plan de requête : lorsqu'une requête est soumise à Neptune, la chaîne de requête est analysée, optimisée et transformée en un plan de requête, qui est ensuite exécuté par le moteur. Les applications sont souvent soutenues par des modèles de requête courants instanciés avec des valeurs différentes. Le cache des plans de requêtes peut réduire la latence globale en mettant en cache les plans de requêtes et en évitant ainsi l'analyse et l'optimisation pour de tels modèles répétés. Pour plus d'informations, consultez Cache du plan de requête dans Amazon Neptune.
- Amélioration des performances pour les requêtes d'agrégation DISTINCTES.
- Amélioration des performances pour les jointures impliquant des variables nullables.
- Amélioration des performances pour les requêtes impliquant un prédicat non égal au prédicat id (node/relation).
- Support étendu pour la fonctionnalité date/heure (uniquement activée en mode laboratoire `DatetimeMillisecond` en incluant`DatetimeMillisecond=enabled`. Pour de plus amples informations, veuillez consulter Support temporel dans l'implémentation de Neptune OpenCypher (Neptune Analytics et Neptune Database 1.3.2.0 et versions ultérieures).

Corrections de défauts reportées depuis la version 1.3.2.0 du moteur

Améliorations générales

- Le message d'erreur NeptuneML a été mis à jour lors de la validation de l'accès aux compartiments Graphlytics.

Correctifs apportés à Gremlin

- Correction d'informations d'étiquette manquantes dans la traduction des requêtes DFE, pour les scénarios dans lesquels les étapes ne contribuant pas au chemin contiennent des étiquettes. Par exemple :

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'marko').
  has("name", TextP.regex("mark.*")).as("p1").
  not(out().has("name", P.within("peter"))).
  out().as('p2').
  dedup('p1', 'p2')
```

- Correction d'un `NullPointerException` bogue dans la traduction des requêtes DFE, qui se produit lorsqu'une requête est exécutée en deux fragments DFE et que le premier fragment est optimisé pour un nœud insatisfaisant. Par exemple :

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'doesNotExists').
  has("name", TextP.regex("mark.*")).
  inject(1).
  V().
  out().
  has('name', 'vadas')
```

- Correction d'un bug en raison duquel Neptune pouvait lancer un modulateur `InternalFailureException` lorsqu'une requête contenait un modulateur ValueTraversal inside by () et que son entrée était Map. Par exemple :

```
g.V().
  hasLabel("person").
  project("age", "name").by("age").by("name").
  order().by("age")
```

Correctifs apportés à openCypher

- Amélioration des opérations UNWIND (par exemple, extension d'une liste de valeurs en valeurs individuelles) pour éviter les situations de manque de mémoire (OOM). Par exemple :

```
MATCH (n)-->(m)
WITH collect(m) AS list
UNWIND list AS m
RETURN m, list
```

- Optimisation de l'identifiant personnalisé fixe en cas d'opérations MERGE multiples où l'identifiant est injecté via UNWIND. Par exemple :

```
UNWIND [{nid: 'nid1', mid: 'mid1'}, {nid: 'nid2', mid: 'mid2'}] as ids
MERGE (n:N {`~id`: ids.nid})
MERGE (m:M {`~id`: ids.mid})
```

- Correction d'une explosion de mémoire lors de la planification de requêtes complexes avec accès aux propriétés et de sauts multiples associés à des relations bidirectionnelles. Par exemple :

```
MATCH (person1:person)-[:likes]->(res)-[:partOf]->(group)-[:knows]-(:entity {name:
 'foo'}),
      (person1)-[:knows]->(person2)-[:likes]-(res2), (comment)-[:presentIn]->(:Group
 {name: 'barGroup'}),
      (person1)-[:commented]->(comment2:comment)-[:partOf]->(post:Post), (comment2)-
[:presentIn]->(:Group {name: 'fooGroup'}),
      (comment)-[:contains]->(info:Details)-[:CommentType]->(:CommentType {name:
 'Positive'}),
      (comment2)-[:contains]->(info2:Details)-[:CommentType]->(:CommentType {name:
 'Positive'})
WHERE datetime('2020-01-01T00:00') <= person1.addedAfter <=
 datetime('2023-01-01T23:59') AND comment.approvedBy = comment2.approvedBy
MATCH (comment)-[:contains]->(info3:Details)-[:CommentType]->(:CommentType {name:
 'Neutral'})
RETURN person1, group.name, info1.value,  post.ranking, info3.value
```

- Requêtes d'agrégation fixes avec null en tant que groupe par variables. Par exemple :

```
MATCH (n)
RETURN null AS group, sum(n.num) AS result
```

Correctifs apportés à SPARQL

- Correction de l'analyseur SPARQL afin d'améliorer le temps d'analyse pour les requêtes volumineuses telles que INSERT DATA contenant de nombreux triplets et de gros jetons.

Atténuation des problèmes dans la version 1.3.2.1

- Les requêtes utilisant des valeurs de filtre numériques peuvent renvoyer des résultats incorrects lors de l'utilisation du cache du plan de requêtes. Pour éviter ce problème, utilisez l'indice de requête `QUERY:PLANCACHE "disabled"` pour ignorer le cache du plan de requête. Par exemple, utilisez :

```
USING QUERY:PLANCACHE "disabled"
MATCH (n:person)
WHERE n.yearOfBirth > $year
RETURN n

parameters={"year":1950}
```

- Les requêtes utilisant le même nom de paramètre plusieurs fois peuvent échouer avec l'erreur`Parameter name should not be a number and/or contain _internal_ or _modified_user_ string within it. These are reserved for planCache. Otherwise, rerun with HTTP parameter planCache=disabled`. Ignorez le cache du plan de requête comme ci-dessus dans de tels cas ou dupliquez les paramètres comme dans cet exemple :

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$minutes RETURN n

parameters={"minutes":130}
```

  Utilisez le conseil `QUERY:PLANCACHE "disabled"` ou modifiez les paramètres :

```
MATCH (n:movie) WHERE n.runtime>=$rt_min RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$dur_min RETURN n

parameters={"rt_min":130, "dur_min":130}
```

- Les requêtes exécutées avec le protocole Bolt peuvent produire des résultats incorrects s'il s'agit d'une requête UNION ou UNION ALL. Pour éviter ce problème, envisagez d'exécuter la requête en question avec le point de terminaison HTTP. Vous pouvez également exécuter chaque partie de l'union séparément lorsque vous utilisez le protocole Bolt.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.3.2.1, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.3.2.1 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.2.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.2.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.3.2.0 du moteur Amazon Neptune (10/06/2021)

Depuis le 10/06/2021, la version 1.3.2.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> La version 1.3.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version du moteur antérieure à la version 1.3.0.0

ou supérieure, vous devez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres. `neptune1.3` Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1` ou `neptune1.2`, lesquels ne sont pas compatibles avec les versions 1.3.0.0 et supérieures. De même, vous devez utiliser des groupes de paramètres de cluster 1.4.0.0 pour les versions de moteur 1.4.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

⚠ Warning

La version 1.3.2.0 du moteur a introduit des problèmes potentiels dont vous devez être conscient. Consultez la section ci-dessous Atténuation des problèmes dans la version 1.3.2.0 pour plus d'informations.

Améliorations apportées à cette version du moteur

Améliorations générales

- Support de la version 1.3 de TLS, y compris les suites de chiffrement TLS_AES_128_GCM_ et TLS_AES_256_GCM_SHA256 . SHA384 Le protocole TLS 1.3 est une option, le protocole TLS 1.2 reste le minimum.

Améliorations apportées à Gremlin

- TinkerPop Mise à niveau 3.7.x
  - Fournit une extension importante du langage Gremlin.
    - Nouvelles étapes pour le traitement des chaînes, des listes et des dates.
    - Nouvelle syntaxe pour spécifier la cardinalité au sein de l'`mergeV()`étape.
    - `union()`peut désormais être utilisé comme étape de départ.
    - Pour en savoir plus sur les modifications apportées à la version 3.7.x, consultez la documentation de TinkerPop mise à niveau.
  - Lors de la mise à niveau des pilotes de langage Gremlin du client pour Java, notez que les classes du sérialiseur n'ont pas été renommées. Vous devrez mettre à jour le nom des packages et des classes dans vos fichiers de configuration et dans le code, si cela est spécifié.

- `StrictTimeoutValidation`(uniquement lorsqu'il est activé via labmode `StrictTimeoutValidation` en incluant`StrictTimeoutValidation=enabled`) : Lorsque le `StrictTimeoutValidation` paramètre a une valeur de`enabled`, une valeur de délai par requête spécifiée en tant qu'option de demande ou indice de requête ne peut pas dépasser la valeur définie globalement dans le groupe de paramètres. Dans ce cas, Neptune lancera un. `InvalidParameterException` Ce paramètre peut être confirmé dans une réponse sur le `/status` terminal lorsque la valeur est`disabled`, et dans Neptune version 1.3.2.0, la valeur par défaut de ce paramètre est. `Disabled`

Améliorations d'openCypher

- La version 1.3.2.0 du moteur Amazon Neptune fournit un débit jusqu'à 9 fois plus rapide et 10 fois plus élevé pour les performances des requêtes OpenCypher par rapport aux versions précédentes du moteur.
- Requêtes à faible latence et amélioration des performances de débit : améliorations des performances globales pour les requêtes OpenCypher à faible latence. La nouvelle version améliore également le débit de ces requêtes. Les améliorations sont plus importantes lorsque des requêtes paramétrées sont utilisées.
- Support pour le cache du plan de requête : lorsqu'une requête est soumise à Neptune, la chaîne de requête est analysée, optimisée et transformée en un plan de requête, qui est ensuite exécuté par le moteur. Les applications sont souvent soutenues par des modèles de requête courants instanciés avec des valeurs différentes. Le cache des plans de requêtes peut réduire la latence globale en mettant en cache les plans de requêtes et en évitant ainsi l'analyse et l'optimisation pour de tels modèles répétés.
- Amélioration des performances pour les requêtes d'agrégation DISTINCTES.
- Amélioration des performances pour les jointures impliquant des variables nullables.
- Amélioration des performances pour les requêtes impliquant un prédicat non égal au prédicat id (node/relation).
- Support étendu pour la fonctionnalité date/heure (uniquement activée en mode laboratoire `DatetimeMillisecond` en incluant`DatetimeMillisecond=enabled`. Pour de plus amples informations, veuillez consulter Support temporel dans l'implémentation de Neptune OpenCypher (Neptune Analytics et Neptune Database 1.3.2.0 et versions ultérieures).

Défauts corrigés dans cette version du moteur

Améliorations générales

- Le message d'erreur NeptuneML a été mis à jour lors de la validation de l'accès aux compartiments Graphlytics.

Correctifs apportés à Gremlin

- Correction d'informations d'étiquette manquantes dans la traduction des requêtes DFE, pour les scénarios dans lesquels les étapes ne contribuant pas au chemin contiennent des étiquettes. Par exemple :

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'marko').
  has("name", TextP.regex("mark.*")).as("p1").
  not(out().has("name", P.within("peter"))).
  out().as('p2').
  dedup('p1', 'p2')
```

- Correction d'un `NullPointerException` bogue dans la traduction des requêtes DFE, qui se produit lorsqu'une requête est exécutée en deux fragments DFE et que le premier fragment est optimisé pour un nœud insatisfaisant. Par exemple :

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'doesNotExists').
  has("name", TextP.regex("mark.*")).
  inject(1).
  V().
  out().
  has('name', 'vadas')
```

- Correction d'un bug en raison duquel Neptune pouvait lancer un modulateur `InternalFailureException` lorsqu'une requête contenait un modulateur ValueTraversal inside by () et que son entrée était Map. Par exemple :

```
g.V().
  hasLabel("person").
  project("age", "name").by("age").by("name").
```

```
  order().by("age")
```

Correctifs apportés à openCypher

- Amélioration des opérations UNWIND (par exemple, extension d'une liste de valeurs en valeurs individuelles) pour éviter les situations de manque de mémoire (OOM). Par exemple :

```
MATCH (n)-->(m)
WITH collect(m) AS list
UNWIND list AS m
RETURN m, list
```

- Optimisation de l'identifiant personnalisé fixe en cas d'opérations MERGE multiples où l'identifiant est injecté via UNWIND. Par exemple :

```
UNWIND [{nid: 'nid1', mid: 'mid1'}, {nid: 'nid2', mid: 'mid2'}] as ids
MERGE (n:N {`~id`: ids.nid})
MERGE (m:M {`~id`: ids.mid})
```

- Correction d'une explosion de mémoire lors de la planification de requêtes complexes avec accès aux propriétés et de sauts multiples associés à des relations bidirectionnelles. Par exemple :

```
MATCH (person1:person)-[:likes]->(res)-[:partOf]->(group)-[:knows]-(:entity {name:
 'foo'}),
      (person1)-[:knows]->(person2)-[:likes]-(res2), (comment)-[:presentIn]->(:Group
 {name: 'barGroup'}),
      (person1)-[:commented]->(comment2:comment)-[:partOf]->(post:Post), (comment2)-
[:presentIn]->(:Group {name: 'fooGroup'}),
      (comment)-[:contains]->(info:Details)-[:CommentType]->(:CommentType {name:
 'Positive'}),
      (comment2)-[:contains]->(info2:Details)-[:CommentType]->(:CommentType {name:
 'Positive'})
WHERE datetime('2020-01-01T00:00') <= person1.addedAfter <=
 datetime('2023-01-01T23:59') AND comment.approvedBy = comment2.approvedBy
MATCH (comment)-[:contains]->(info3:Details)-[:CommentType]->(:CommentType {name:
 'Neutral'})
RETURN person1, group.name, info1.value,  post.ranking, info3.value
```

- Requêtes d'agrégation fixes avec null en tant que groupe par variables. Par exemple :

```
MATCH (n)
```

```
RETURN null AS group, sum(n.num) AS result
```

Correctifs apportés à SPARQL

- Correction de l'analyseur SPARQL afin d'améliorer le temps d'analyse pour les requêtes volumineuses telles que INSERT DATA contenant de nombreux triplets et de gros jetons.

Atténuation des problèmes dans la version 1.3.2.0

- Pour la version 1.3.2.0, nous avons détecté un problème dans le cache du plan de requête lorsque `skip` ou `limit` est utilisé dans une `WITH` clause interne et qu'il est paramétré. Par exemple :

  ```
  MATCH (n:Person)
  WHERE n.age > $age
  WITH n skip $skip LIMIT $limit
  RETURN n.name, n.age

  parameters={"age": 21, "skip": 2, "limit": 3}
  ```

  Dans ce cas, les valeurs des paramètres pour skip et limit du premier plan seront également appliquées aux requêtes suivantes, ce qui entraînera des résultats inattendus.

  Mitigation

  Pour éviter ce problème, ajoutez l'indice de requête `QUERY:PLANCACHE "disabled"` lorsque vous soumettez une requête qui inclut une sous-clause de and/or limite de saut paramétrée. Vous pouvez également coder les valeurs en dur dans la requête.

  Option 1 : utilisation de l'indicateur de requête pour désactiver le cache du plan :

  ```
  Using QUERY:PLANCACHE "disabled"
  MATCH (n:Person) WHERE n.age > $age
  WITH n skip $skip LIMIT $limit
  RETURN n.name, n.age

  parameters={"age": 21, "skip": 2, "limit": 3}
  ```

  Option 2 : utilisation de valeurs codées en dur pour le saut et la limite :

```
MATCH (n:Person)
WHERE n.age > $age
WITH n skip 2 LIMIT 3
RETURN n.name, n.age

parameters={"age": 21}
```

- Les requêtes utilisant des valeurs de filtre numériques peuvent renvoyer des résultats incorrects lors de l'utilisation du cache du plan de requêtes. Pour éviter ce problème, utilisez l'indice de requête `QUERY:PLANCACHE "disabled"` pour ignorer le cache du plan de requête. Par exemple, utilisez :

```
USING QUERY:PLANCACHE "disabled"
MATCH (n:person)
WHERE n.yearOfBirth > $year
RETURN n

parameters={"year":1950}
```

- Les requêtes utilisant le même nom de paramètre plusieurs fois peuvent échouer avec l'erreur`Parameter name should not be a number and/or contain _internal_ or _modified_user_ string within it. These are reserved for planCache. Otherwise, rerun with HTTP parameter planCache=disabled`. Ignorez le cache du plan de requête comme ci-dessus dans de tels cas ou dupliquez les paramètres comme dans cet exemple :

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$minutes RETURN n

parameters={"minutes":130}
```

  Utilisez le conseil `QUERY:PLANCACHE "disabled"` ou modifiez les paramètres :

```
MATCH (n:movie) WHERE n.runtime>=$rt_min RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$dur_min RETURN n

parameters={"rt_min":130, "dur_min":130}
```

- Les requêtes exécutées avec le protocole Bolt peuvent produire des résultats incorrects s'il s'agit d'une requête UNION ou UNION ALL. Pour éviter ce problème, envisagez d'exécuter la requête en question avec le point de terminaison HTTP. Vous pouvez également exécuter chaque partie de l'union séparément lorsque vous utilisez le protocole Bolt.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.3.2.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.7.1`
- Dernière version de Gremlin est prise en charge : `3.7.1`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.3.2.0 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.2.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.2.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.3.1.0 du moteur Amazon Neptune (06/03/2021)

Depuis le 06/03/2021, la version 1.3.1.0 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> La version 1.3.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version du moteur antérieure à la version 1.3.0.0 ou supérieure, vous devez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres. `neptune1.3` Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1` ou `neptune1.2`, lesquels ne sont pas compatibles avec les versions 1.3.0.0 et supérieures. De même, vous devez utiliser des groupes de paramètres de cluster 1.4.0.0 pour les versions de moteur 1.4.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

Améliorations apportées à cette version du moteur

Améliorations générales

- Neptune a amélioré l'avertissement affiché dans profile/explain.
- Suppression des courbes NIST EC obsolètes des groupes nommés par défaut utilisés lors de la négociation TLS. Les courbes supprimées sont sect409k1, sect409r1 et sect571k1.

Améliorations apportées à Gremlin

- Amélioration du calcul des statistiques DFE pour éviter un nombre très élevé NCUs d'instances sans serveur.
- Amélioration des performances de G705 pour WITHIN.

Défauts corrigés dans cette version du moteur

Correctifs apportés à Gremlin

- Améliorations diverses apportées aux plans de requêtes Gremlin DFE.
- Correction d'un bogue pour les requêtes Gremlin avec une traversée facultative, par exemple pour les requêtes de la forme « g.V () .hasLabel ('person') .group () .by (id ()) .by (__.in ('friend') .id () .fold ()) `, où aucune personne n'ayant pas d'avantage d'amis n'était groupée.
- Correction d'un bug à cause duquel les requêtes G705 contenant des étapes de fusion à l'intérieur des by modulateurs provoquaient le renvoi d'une erreur si elles étaient exécutées à l'aide du moteur DFE.
- Correction d'un bogue qui empêchait les requêtes en lecture seule exécutées dans une session G705 de fonctionner lorsqu'elles étaient connectées à une réplique en lecture.
- Correction d'un bogue en raison duquel l'ARN IAM n'était pas présent dans le journal d'audit en raison d'une demande initiale de connexion Websocket réussie pour G705.
- Étape de fusion, identification de la couverture des étapes avec DFE.
- Optimisation des ensembles de caractéristiques pour l'ensemble des plans DFE.

Correctifs apportés à openCypher

- Corrections de bogues dans la clause OpenCypher SET pour autoriser le réglage sur une expression non variable (par exemple : match (n:Test) set (cas où n.prop = 2 puis n end) .prop = 3 renvoie n.prop.
- Correction d'un bogue concernant l'échec des requêtes OpenCypher impliquant l'agrégation et le tri par.
- Amélioration de la fonction UNWIND d'une grande liste contenant des cartes statiques.
- Correction d'une requête OpenCypher MERGE utilisant un identifiant personnalisé avec des valeurs dupliquées.

Correctifs apportés à SPARQL

- Correction d'un bogue SPARQL concernant la portée de la variable dans les modèles optionnels.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.3.1.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.6.2`
- Dernière version de Gremlin est prise en charge : `3.6.5`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.3.1.0 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.3.0.0 du moteur Amazon Neptune (15-11-2023)

Depuis le 15-11-2023, la version 1.3.0.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> La version 1.3.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version du moteur antérieure à la version 1.3.0.0 ou supérieure, vous devez recréer tous vos groupes de paramètres personnalisés et

groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres. `neptune1.3` Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1` ou `neptune1.2`, lesquels ne sont pas compatibles avec les versions 1.3.0.0 et supérieures. De même, vous devez utiliser des groupes de paramètres de cluster 1.4.0.0 pour les versions de moteur 1.4.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

Nouvelles fonctionnalités pour cette version du moteur

- Publication de l'API de données Neptune.

  L'API de données Amazon Neptune fournit un support de kit SDK pour plus de 40 opérations de données de Neptune, notamment le chargement des données, l'exécution de requêtes, la recherche de données et le machine learning. Elle prend en charge les trois langages de requête Neptune (Gremlin, openCypher et SPARQL) et est disponible dans tous les langages de kit SDK. Elle signe automatiquement les demandes d'API et simplifie considérablement l'intégration de Neptune dans les applications.
- Ajout de la prise en charge de l'intégration de OpenSearchServerless à Neptune.

Améliorations de cette version du moteur

Améliorations apportées aux mises à jour du moteur Neptune

Neptune a modifié la façon dont il publie les mises à jour du moteur afin que vous puissiez mieux contrôler le processus de mise à jour. Au lieu de publier des correctifs pour des modifications non majeures, Neptune publie désormais des versions mineures qui peuvent être contrôlées à l'aide du champ d'instance `AutoMinorVersionUpgrade` et pour lesquelles vous pouvez recevoir des notifications en vous abonnant à l'événement `RDS-EVENT-0156`.

Consultez Maintenance du cluster de bases de données Amazon Neptune pour plus d'informations sur ces modifications.

Amélioration du chiffrement en transit

Neptune ne prend plus en charge les suites de chiffrement suivantes :

- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`

- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

Neptune ne prend en charge que les suites de chiffrement puissantes suivantes avec TLS 1.2 :

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`

Améliorations de Gremlin

- Ajout de support dans le moteur DFE pour les étapes Gremlin suivantes :
  - `FoldStep`
  - `GroupStep`
  - `GroupCountStep`
  - `TraversalMapStep`
  - `UnfoldStep`
  - `LabelStep`
  - `PropertyKeyStep`
  - `PropertyValueStep`
  - `AndStep`
  - `OrStep`
  - `ConstantStep`
  - `CountGlobalStep`
- Plans de requête DFE optimisés pour éviter les analyses complètes des sommets lors de l'utilisation de la modulation `by()`.
- Amélioration significative des performances des requêtes à faible cardinalité et à faible latence.
- Ajout du support DFE pour les prédicats de TinkerPop `Or` filtre.
- Support DFE amélioré de la traversée pour les filtres sur la même clé, pour des requêtes telles que les suivantes :

```
g.withSideEffect("Neptune#useDFE", true)
 .V()
```

```
  .has('name', 'marko')
  .and(
    or(
      has('name', eq("marko")),
      has('name', eq("vardas"))
    )
  )
```

- Gestion des erreurs améliorée pour l'étape `fail()`.

Améliorations d'openCypher

- Amélioration significative des performances des requêtes à faible cardinalité et à faible latence.
- Amélioration des performances de planification des requêtes lorsque la requête contient de nombreux types de nœuds.
- Latence réduite de toutes les requêtes VLP.
- Performances améliorées en supprimant les jointures de pipeline redondantes pour les requêtes de modèle à nœud unique.
- Performances améliorées pour les requêtes contenant des modèles à sauts multiples avec des cycles, comme celui-ci :

```
MATCH (n)-->()-->()-->(m)
RETURN n m
```

Améliorations de SPARQL

- Ajout d'un nouvel opérateur SPARQL : `PipelineHashIndexJoin`.
- Amélioration des performances de validation d'URI pour les requêtes SPARQL.
- Amélioration des performances des requêtes de recherche en texte intégral SPARQL grâce à la résolution par lots des termes du dictionnaire.

Défauts corrigés dans cette version du moteur

Correctifs apportés à Gremlin

- Correction d'un bogue Gremlin qui provoquait une fuite de transaction lors de la vérification du point de terminaison de statut d'une requête Gremlin pour détecter les requêtes contenant des

prédicats dans les traversées enfants pour les étapes qui ne sont pas traitées nativement dans le moteur DFE.

- Correction d'un bogue Gremlin qui empêchait l'optimisation de `valueMap()` lors de traversées `by()` dans le moteur DFE.
- Correction d'un bogue Gremlin qui empêchait la propagation d'une étiquette d'étape attachée à `UnionStep` au dernier élément de chemin de ses traversées enfants.
- Correction d'un bogue Gremlin qui empêchait une requête d'échouer parce qu'elle contenait trop d' TinkerPop étapes et ne pouvait donc pas être nettoyée.
- Correction d'un bug de Gremlin qui provoquait l'ajout d'une `NullPointerException` dans les étapes `mergeV` et `mergeE`.
- Correction d'un bogue Gremlin en raison duquel `order()` empêchait de trier correctement les sorties de chaînes lorsque certaines d'entre elles contenaient un espace.
- Correction d'un problème d'exactitude de Gremlin qui se produisait lorsque l'étape `valueMap` était traitée dans le moteur DFE.
- Correction d'un problème d'exactitude de Gremlin qui se produisait lorsque `GroupStep` ou `GroupCountStep` était imbriqué dans une traversée de touches.

Correctifs apportés à openCypher

- Correction d'un bogue openCypher impliquant la gestion des erreurs autour des caractères NULL.
- Correction d'un bogue openCypher lié à la gestion des transactions Bolt.

Correctifs apportés à SPARQL

- Correction d'un bogue SPARQL en raison duquel les valeurs contenues dans les fonctions récursives n'étaient pas correctement résolues.
- Correction d'un bogue SPARQL en raison duquel un grand nombre de valeurs injectées via une clause `VALUES` pouvait entraîner une dégradation des performances.
- Correction d'un bogue SPARQL qui empêchait l'opérateur `REGEX` d'aboutir lorsqu'il était appelé sur un littéral balisé dans un langage.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.3.0.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.6.2`
- Dernière version de Gremlin est prise en charge : `3.6.4`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.3.0.0 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.1.2 du moteur Amazon Neptune (2024-08-05)

Depuis le 5 août 2021, la version 1.2.1.2 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :
>
> - La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent,

si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau depuis une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

  Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

  Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.
- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

> **Warning**
>
> Un problème a été détecté avec le protocole HTTP SPARQL 1.1 Graph Store (GSP) qui peut être présent dans certaines conditions lorsque le GSP est utilisé avec des politiques d'autorisation basées sur l'action. Si vous utilisez le protocole HTTP SPARQL 1.1 Graph Store avec des politiques d'autorisation basées sur des actions, nous vous recommandons de passer à la dernière version mineure du moteur Neptune (au moins 1.2.1.2) qui inclut un correctif pour ce problème.

Défauts corrigés dans cette version du moteur

Correctifs apportés à SPARQL

- Correction d'un problème lié au protocole HTTP SPARQL 1.1 Graph Store (GSP) qui pouvait être présent dans certaines conditions lorsque le GSP est utilisé avec des politiques d'autorisation basées sur des actions.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.2.1.2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.6.2`
- Dernière version de Gremlin est prise en charge : `3.6.2`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.2.1.2 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez

mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.2 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.2 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise

à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the instance before
>   proceeding with the upgrade.
> ```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.1.1 du moteur Amazon Neptune (2024-03-11)

Depuis le 11/03/2021, la version 1.2.1.1 du moteur est généralement déployée. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Note

En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :

- La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau depuis une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.

- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

Améliorations générales

Neptune a amélioré l'avertissement affiché dans profile/explain.

Améliorations de Gremlin

- Amélioration du calcul des statistiques DFE pour éviter un nombre très élevé NCUs d'instances sans serveur.
- Amélioration des performances de Gremlin pour WITHIN.

Défauts corrigés dans cette version du moteur

Correctifs apportés à Gremlin

- Corrections de bogues concernant la commande du plan de requêtes du moteur Gremlin DFE.
- Correction d'un bogue avec une out-of-memory erreur Gremlin signalée à l'origine sous la forme InternalFailureException.

- Correction d'un bogue en raison duquel l'ARN IAM n'était pas présent dans le journal d'audit en raison d'une demande initiale de connexion Websocket réussie.
- Correction d'un bogue pour les requêtes Gremlin avec TinkerPop session activée lorsque les requêtes d'une session échouent, même si elles sont toutes en lecture seule et se connectent à une instance de lecteur.

Correctifs apportés à openCypher

- Corrections de bogues dans la clause OpenCypher SET pour autoriser le réglage sur une expression non variable (par exemple : match (n:Test) set (cas où n.prop = 2 puis n end) .prop = 3 renvoie n.prop.
- Correction d'un bogue concernant l'échec des requêtes OpenCypher impliquant l'agrégation et le tri par.
- Amélioration de la fonction UNWIND d'une grande liste contenant des cartes statiques.
- Correction d'un bogue dans la requête OpenCypher MERGE utilisant un identifiant personnalisé avec des valeurs dupliquées.

Correctifs apportés à SPARQL

- Corrections de bogues dans le planificateur de requêtes SPARQL DFE.
- Correction d'un bogue pour SPARQL lorsqu'il est utilisé avec les mots clés BIND et OPTIONAL.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de base de données vers la version 1.2.1.1, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.6.2`
- Dernière version de Gremlin est prise en charge : `3.6.2`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version 1.2.1.1 du moteur

Vous pouvez effectuer une mise à niveau vers cette version à partir de la version 1.2.0.0 ou supérieure du moteur.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.0 du moteur Amazon Neptune (08/03/2023)

Depuis le 8 mars 2023, la version 1.2.1.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :
>
> - La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles

avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

  Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

  Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.
- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Versions de correctifs ultérieures pour cette version

- Sortie : 1.2.1.0.R2 (02/05/2023)
- Sortie : 1.2.1.0.R3 (13/06/2023)
- Sortie : 1.2.1.0.R4 (10/08/2023)
- Sortie : 1.2.1.0.R5 (02/09/2023)
- Sortie : 1.2.1.0.R6 (12/09/2023)

- Version : 1.2.1.0.R7 (06-10-2023)

Nouvelles fonctionnalités pour cette version du moteur

- Ajout du support pour la TinkerPop version 3.6.2, qui ajoute de nombreuses nouvelles fonctionnalités de Gremlin, telles que les nouveaux`mergeV()`, `mergeE()element()`, et les `fail()` étapes. Les étapes `mergeV()` et `mergeE()` sont particulièrement intéressantes, car elles offrent une option déclarative tant attendue pour effectuer des opérations de type upsert. Cela devrait grandement simplifier les modèles de code existants et faciliter la lecture de Gremlin. La version 3.6.x a également ajouté des prédicats regex (expression régulière), une nouvelle surcharge à l'étape `property()` qui utilise un mappage (`Map`), ainsi qu'une révision majeure du comportement de modulation `by()` qui est beaucoup plus cohérente dans toutes les étapes qui l'utilisent.

  Consultez le journal des TinkerPop modifications et la page de mise à niveau pour obtenir des informations sur les modifications apportées à la version 3.6 et les éléments à prendre en compte lors de la mise à niveau.

  Si vous utilisez `fold().coalesce(unfold(), <mutate>)` pour les insertions conditionnelles, nous vous recommandons de migrer vers la nouvelle syntaxe `mergeV/E()`, décrite ici et ici. Neptune utilise un schéma de verrouillage plus étroit pour `Merge` que pour`Coalesce`, ce qui permet de réduire les exceptions de modification simultanées (). CMEs

  Pour plus d'informations sur les nouvelles fonctionnalités disponibles dans cette TinkerPop version, consultez le blog de Stephen Mallette, Exploring new features of Apache TinkerPop 3.6.x in Amazon Neptune.
- Ajout de la prise en charge des types d'instances R6i, alimentés par des processeurs Intel Xeon Scalable de 3e génération. Elles sont parfaitement adaptées aux charges de travail gourmandes en mémoire et offrent des compute/price performances jusqu'à 15 % supérieures et une bande passante mémoire jusqu'à 20 % supérieure par vCPU par rapport aux types d'instances R5 comparables.
- Ajout de points de terminaison d'API de résumé de graphe pour les graphes de propriétés et les graphes RDF, afin de vous permettre de générer un rapport récapitulatif rapide du graphe.

  Pour les graphes de propriétés (PG), l'API de résumé de graphe fournit une liste en lecture seule des étiquettes et des clés de propriété des nœuds et des arêtes, ainsi que le nombre de nœuds,

d'arêtes et de propriétés. Pour les graphes RDF, elle fournit une liste de classes et de clés de prédicat, ainsi que le nombre de quadruplets, de sujets et de prédicats.

Les modifications suivantes ont été apportées avec la nouvelle API de résumé de graphe :

- Ajout d'une nouvelle action sur le GetGraphSummaryplan de données.
- Ajout d'un nouveau point de terminaison `rdf/statistics` pour remplacer le point de terminaison `sparql/statistics`, qui est désormais obsolète.
- Remplacement du nom du champ `summary` dans la réponse d'état des statistiques par `signatureInfo`, afin de ne pas le confondre avec les informations du résumé de graphe. Les versions précédentes du moteur continueront d'utiliser `summary` dans la réponse JSON.
- La précision du champ `date` dans la réponse d'état des statistiques a été modifiée, passant de la minute à la milliseconde. Le format précédent était `2020-05-07T23:13Z` (précision à la minute), tandis que le nouveau format est `2023-01-24T00:47:43.319Z` (précision à la milliseconde). Ces deux formats sont conformes à la norme ISO 8601, mais cette modification peut corrompre le code existant, en fonction de la façon dont la date est analysée.
- Une nouvelle magie linéaire `%statistics`, qui vous permet de récupérer les statistiques du moteur DFE, a été ajoutée dans le workbench.
- Une nouvelle magie linéaire `%summary`, qui vous permet de récupérer les informations du résumé de graphe, a été ajoutée dans le workbench.

- Ajout de la journalisation des requêtes lentes pour enregistrer les requêtes dont l'exécution prend plus de temps qu'un seuil spécifié. Vous devez activer et contrôler la journalisation des requêtes lentes à l'aide des deux nouveaux paramètres dynamiques, à savoir neptune_enable_slow_query_log et neptune_slow_query_log_threshold.
- Ajout de la prise en charge de deux paramètres dynamiques, à savoir les nouveaux paramètres de cluster neptune_enable_slow_query_log et neptune_slow_query_log_threshold. Lorsque vous modifiez un paramètre dynamique, il s'applique immédiatement, sans nécessiter de redémarrage de l'instance.
- Ajout d'une fonction OpenCypher removeKeyFromMap () spécifique à Neptune qui supprime une clé spécifiée d'une carte et renvoie la nouvelle carte résultante.

Améliorations de cette version du moteur

- Prise en charge étendue de Gremlin DFE pour les étapes `limit` avec une portée locale.

- Ajout de la prise en charge de la modulation `by()` pour l'étape Gremlin `DedupGlobalStep` dans le moteur DFE.
- Ajout de la prise en charge DFE des étapes Gremlin `SelectStep` et `SelectOneStep`.
- Amélioration des performances et corrections de divers opérateurs Gremlin, notamment `repeat`, `coalesce`, `store` et `aggregate`.
- Amélioration des performances des requêtes openCypher impliquant `MERGE` et `OPTIONAL MATCH`.
- Amélioration des performances des requêtes openCypher impliquant `UNWIND` pour une liste de mappages de valeurs littérales.
- Amélioration des performances des requêtes openCypher dotées d'un filtre `IN` pour `id`. Par exemple :

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Ajout de la possibilité de spécifier l'IRI de base pour les requêtes SPARQL à l'aide de l'instruction BASE (voir IRI de base par défaut pour les requêtes et les mises à jour).
- Réduction du temps d'attente pour le traitement des chargements en bloc spécifiques aux arêtes Gremlin et openCypher.
- Les chargements en bloc reprennent désormais de manière asynchrone lorsque Neptune redémarre afin d'éviter un long délai d'attente causé par des problèmes de connectivité à Amazon S3 avant l'échec des tentatives de reprise.
- Gestion améliorée des requêtes SPARQL DESCRIBE dont l'indicateur de requête DescribeMode est défini sur `"CBD"` (description CBD) et qui impliquent un grand nombre de nœuds vides.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue openCypher où les requêtes renvoyaient la chaîne `"null"` au lieu d'une valeur nulle dans Bolt et SPARQL-JSON.
- Correction d'un bogue openCypher lié à la compréhension de liste, où une valeur nulle était générée au lieu des valeurs fournies pour les éléments de liste.
- Correction d'un bogue openCypher en raison duquel les valeurs d'octets n'étaient pas correctement sérialisées.
- Correction d'un bogue Gremlin dans l'étape `UnionStep` qui se produisait lorsqu'une entrée était une arête traversante vers un sommet dans une traversée enfant.

- Correction d'un bogue Gremlin qui empêchait la propagation correcte d'une étiquette d'étape associée à `UnionStep` vers la dernière étape de chaque traversée enfant.
- Correction d'un bogue Gremlin pour l'étape `dedup` avec des étiquettes suivant une étape `repeat`, bogue à cause duquel les étiquettes attachées à l'étape `dedup` n'étaient pas disponibles pour une utilisation ultérieure dans la requête.
- Correction d'un bogue Gremlin qui empêchait la conversion d'une étape `repeat` dans une étape `union` en raison d'une erreur interne.
- Correction des problèmes d'exactitude Gremlin pour les requêtes DFE utilisant `limit` comme traversée enfant des étapes autres que l'union en revenant à Tinkerpop. Les requêtes sous la forme suivante sont affectées :

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- Correction d'un bogue SPARQL qui empêchait les modèles `SPARQL GRAPH` de prendre en compte le jeu de données fourni par une clause `FROM NAMED`.
- Correction d'un bogue SPARQL en raison duquel le SPARQL, `DESCRIBE` avec certaines `FROM` and/or `FROM NAMED` clauses, n'utilisait pas toujours correctement les données du graphe par défaut et lançait parfois une exception. Consultez Comportement de SPARQL DESCRIBE par rapport au graphe par défaut.
- Correction d'un bogue SPARQL qui renvoyait le message d'exception correct en cas de rejet de caractères null.
- Correction d'un bogue d'explication dans le SPARQL qui affectait les plans contenant un PipelinedHashIndexJoinopérateur.
- Correction d'un bogue qui provoquait le déclenchement d'une erreur interne lorsqu'une requête renvoyant une valeur de constante était soumise.
- Correction d'un problème lié à la logique du détecteur de blocage qui empêchait parfois le moteur de répondre.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.1.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.6.2`
- Dernière version de Gremlin est prise en charge : `3.6.2`

- Version d'openCypher : `Neptune-9.0.20190305-1.1`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.1.0

Vous pouvez effectuer une mise à niveau manuelle vers cette version à partir de n'importe quelle version de moteur Neptune précédente supérieure ou égale à 1.1.0.0.

> **Note**
>
> À partir de la version 1.2.0.0 du moteur, tous les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés que vous utilisiez avec les versions de moteur antérieures à `1.2.0.0` doivent désormais être recréés à l'aide de la famille `neptune1.2` de groupes de paramètres. Les versions précédentes utilisaient une famille de groupes de paramètres `neptune1`, et ces groupes de paramètres ne fonctionnent pas avec les versions `1.2.0.0` et ultérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

La mise à niveau vers cette version majeure n'est pas automatique.

Mise à niveau vers cette version

Amazon Neptune 1.2.1.0 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un

instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.1.0.R7 du moteur Amazon Neptune (06-10-2023)

Depuis le 06-10-2023, la version 1.2.1.0.R7 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :
>
> - La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de

la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

  Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

  Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.
- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue au cours duquel une transaction ayant échoué n'était pas toujours correctement clôturée.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.1.0.R7, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.6.2`
- Dernière version de Gremlin est prise en charge : `3.6.2`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Mise à niveau vers cette version

Amazon Neptune 1.2.1.0.R7 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
> ```

```
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.1.0.R6 du moteur Amazon Neptune (12/09/2023)

Depuis le 12 septembre 2023, la version 1.2.1.0.R6 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Note

En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :

- La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative

de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.

- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Nouvelles fonctionnalités pour cette version du moteur

- Publication de l'API de données Neptune.

  L'API de données Amazon Neptune fournit une prise en charge de kit SDK pour le chargement de données, l'exécution de requêtes, l'obtention d'informations sur vos données et l'exécution d'opérations de machine learning. Elle prend en charge les langages de requête Gremlin et openCypher dans Neptune et est disponible dans tous les langages de kit SDK. Elle signe automatiquement les demandes d'API et simplifie considérablement l'intégration de Neptune dans les applications.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue qui pouvait provoquer des pics d'activité du CPU sous des charges élevées lorsque les journaux Slow Query étaient activés.

- Correction d'un bogue Gremlin où l'ajout d'une arête et de ses propriétés suivi de `inV()` ou de `outV()` levait une exception `InternalFailureException`.
- Correction de plusieurs problèmes liés au chaînage des rôles IAM, problèmes qui entraînaient une dégradation des performances des chargeurs en bloc dans certains cas.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.1.0.R6, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.6.2`
- Dernière version de Gremlin est prise en charge : `3.6.2`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Mise à niveau vers cette version

Amazon Neptune 1.2.1.0.R6 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.1.0.R5 du moteur Amazon Neptune (02/09/2023)

Depuis le 2 septembre 2023, la version 1.2.1.0.R5 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Note

En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :

- La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch

métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.

- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Nouvelles fonctionnalités pour cette version du moteur

- Publication de l'API de données Neptune.

  L'API de données Amazon Neptune fournit une prise en charge de kit SDK pour le chargement de données, l'exécution de requêtes, l'obtention d'informations sur vos données et l'exécution d'opérations de machine learning. Elle prend en charge les langages de requête Gremlin et openCypher dans Neptune et est disponible dans tous les langages de kit SDK. Elle signe automatiquement les demandes d'API et simplifie considérablement l'intégration de Neptune dans les applications.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin où l'ajout d'une arête et de ses propriétés suivi de `inV()` ou de `outV()` levait une exception `InternalFailureException`.
- Correction de plusieurs problèmes liés au chaînage des rôles IAM, problèmes qui entraînaient une dégradation des performances des chargeurs en bloc dans certains cas.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.1.0.R5, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.6.2`
- Dernière version de Gremlin est prise en charge : `3.6.2`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Mise à niveau vers cette version

Amazon Neptune 1.2.1.0.R5 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.1.0.R4 du moteur Amazon Neptune (10/08/2023)

Depuis le 10 août 2023, la version 1.2.1.0.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Important**
>
> Les modifications apportées à cette version du moteur peuvent, dans certains cas, entraîner une dégradation des performances de chargement en bloc. Par conséquent, les mises à niveau de cette version ont été temporairement suspendues jusqu'à ce que le problème soit résolu.

> **Note**
>
> En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :

- La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

  Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

  Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.
- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

- Ajout de la prise en charge de Graphson-1.0 pour Gremlin. Pour utiliser Graphson-1.0, transmettez `Accept header` avec la valeur :

```
application/vnd.gremlin-v1.0+json;types=false
```

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin qui provoquait une fuite de transaction lors de la vérification du point de terminaison de statut d'une requête Gremlin pour détecter les requêtes contenant des prédicats dans les traversées enfants pour les étapes qui ne sont pas traitées nativement.
- Correction d'un bogue openCypher lié à la gestion des transactions Bolt.
- Correction d'un problème de simultanéité sur la couche de stockage qui pouvait provoquer un plantage.
- Correction d'un bogue dans les journaux de requêtes lentes afin de s'assurer qu'ils ne sont pas actifs lorsqu'ils sont désactivés.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.1.0.R4, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.6.2`
- Dernière version de Gremlin est prise en charge : `3.6.5`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.1.0.R4

Mise à niveau vers cette version

Amazon Neptune 1.2.1.0.R4 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez

mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.1.0.R3 du moteur Amazon Neptune (13/06/2023)

Depuis le 13 juin 2023, la version 1.2.1.0.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Important

Les modifications apportées à cette version du moteur peuvent, dans certains cas, entraîner une dégradation des performances de chargement en bloc. Par conséquent, les mises à niveau de cette version ont été temporairement suspendues jusqu'à ce que le problème soit résolu.

Note

En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :

- La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

  Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer.

L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.

- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Nouvelles fonctionnalités pour cette version du moteur

- Ajout de la prise en charge du chargement en bloc entre comptes à l'aide du chaînage des rôles IAM.

Améliorations de cette version du moteur

- Amélioration de l'étape `fail()` utilisée par Gremlin pour différencier l'exception produite à partir d'une exception `InternalFailureException` générique et pour garantir que tout message fourni par l'utilisateur soit retransmis à l'appelant.
- Amélioration des optimisations du moteur de requêtes Gremlin pour `store`, `aggregate`, `cap`, `limit` et `hasLabel`.
- Ajout de la prise en charge des fonctions trignométriques openCypher :
  - `acos()`
  - `asin()`
  - `atan()`
  - `atan2()`
  - `cos()`
  - `cot()`
  - `degrees()`

  - `pi()`
  - `radians()`
  - `sin()`
  - `tan()`
- Ajout de la prise en charge de plusieurs fonctions d'agrégation openCypher :
  - `percentileDisc()`
  - `stDev()`
- Ajout de la prise en charge de la fonction `epochmillis()` openCypher qui convertit un objet `datetime` en `epochmillis`. Par exemple :

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

- Ajout de la prise en charge de l'opérateur modulo (`%`) openCypher.
- Ajout de la prise en charge de l'outil openCypher Static Debug Explain.
- Ajout de la prise en charge de la fonction openCypher `randomUUID()`.
- Amélioration des performances d'openCypher :
  - Amélioration de l'analyseur et du planificateur de requêtes.
  - Utilisation améliorée du CPU dans le moteur DFE.
  - Amélioration des performances des requêtes contenant plusieurs clauses de mise à jour réutilisant les mêmes variables. Voici quelques exemples :

```
MERGE (n {name: 'John'})
  or
MERGE (m {name: 'Jim'})
  or
MERGE (n)-[:knows {since: 2023}]#(m)
```

  - Plans de requêtes optimisés pour les modèles de requêtes à sauts multiples tels que :

```
MATCH (n)-->()-->()-->(m)
RETURN n m
```

  - Amélioration des performances de l'injection de listes et de mappages grâce à des requêtes paramétrées. Par exemple :

```
UNWIND $idList as id MATCH (n {`~id`: id})
RETURN n.name
```

  - Amélioration de l'exécution des requêtes contenant `WITH` en en faisant une barrière appropriée.
  - Optimisation pour éviter la matérialisation redondante des valeurs `Unfold` et dans les fonctions d'agrégation.
- Amélioration des performances des requêtes SPARQL qui contiennent un grand nombre d'entrées statiques dans la clause `VALUES`, telles que :

```
SELECT ?n WHERE { VALUES (?name) { ("John") ("Jim") ... many values ... } ?n a ?
n_type . ?n ?name . }
```

- Amélioration des performances des requêtes SPARQL CBD.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin à cause duquel les requêtes longues avec imbrication profonde entraînaient une utilisation élevée du CPU et l'expiration des requêtes pendant la phase de planification des requêtes.
- Correction d'un bogue Gremlin où une exception `NullPointerException` non valide pouvait être levée lors de l'utilisation de `mergeV` ou `mergeE`.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.1.0.R3, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.6.2`
- Dernière version de Gremlin est prise en charge : `3.6.2`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.1.0.R3

Mise à niveau vers cette version

Amazon Neptune 1.2.1.0.R3 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations,

l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.1.0.R2 du moteur Amazon Neptune (02/05/2023)

Depuis le 2 mai 2023, la version 1.2.1.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :
>
> - La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
> - La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau depuis une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.
>
>   Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.

- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

- Ajout d'un `enableInterContainerTrafficEncryption` paramètre à tous les Neptune ML APIs, que vous pouvez utiliser pour activer et désactiver le chiffrement du trafic entre conteneurs dans le cadre de tâches d'entraînement ou de réglage d'hyperparamètres.
- Ajout de la prise en charge de plusieurs étiquettes pour les étapes Gremlin `mergeV()` et `mergeE()`.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue openCypher en raison duquel les requêtes de mise à jour et de retour ne géraient pas `orderBy`, `limit` ni `skip` correctement.
- Correction d'un bogue openCypher qui permettait de remplacer les paramètres contenus dans une requête par des paramètres contenus dans une autre requête simultanée.
- Correction d'un bogue openCypher en raison duquel les journaux de requêtes lentes ne contenaient pas les durées correctes des requêtes.
- Correction d'un bogue Gremlin qui provoquait une fuite de transaction lorsqu'une requête contenant `GroupCountStep` était soumise sous forme de chaîne.
- Correction d'un bogue de Gremlin à cause duquel WebSocket les requêtes échouaient lorsque les journaux de requêtes lents étaient activés.
- Correction d'un bogue de Gremlin en raison duquel les journaux de débogage du compteur de stockage manquaient dans les journaux de requêtes lents pour les requêtes. WebSocket
- Correction de plusieurs bogues Gremlin impliquant `mergeV()` et `mergeE()`.

- Correction d'un bogue SPARQL en raison duquel les coûts des requêtes de graphes nommés étaient mal estimés, ce qui entraînait des plans de requêtes sous-optimaux et des erreurs. out-of-memory
- Correction d'un bogue concernant l'autorisation pour les requêtes Gremlin et openCypher sur un cluster compatible IAM.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.1.0.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.6.2`
- Dernière version de Gremlin est prise en charge : `3.6.2`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.1.0.R2

Mise à niveau vers cette version

Amazon Neptune 1.2.1.0.R2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.0.2 du moteur Amazon Neptune (20/11/2022)

Depuis le 20 novembre 2022, la version 1.2.0.2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :
>
> - La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles

avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

  Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

  Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.
- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Versions de correctifs ultérieures pour cette version

- Sortie : 1.2.0.2.R2 (15/12/2022)
- Sortie : 1.2.0.2.R3 (27/03/2023)
- Sortie : 1.2.0.2.R4 (08/05/2023)
- Sortie : 1.2.0.2.R5 (16/08/2023)
- Sortie : 1.2.0.2.R6 (12/09/2023)

Nouvelles fonctionnalités pour cette version du moteur

- Ajout de l'inférence inductive en temps réel pour Gremlin dans Neptune ML.
- Introduction d'une extension OpenCypher qui permet de spécifier des valeurs d'identification personnalisées pour les entités au lieu de celles que UUIDs Neptune génère autrement. La possibilité d'attribuer des options personnalisées IDs facilite la migration de Neo4j vers Neptune.

  > **Warning**
  >
  > Cette extension de la spécification openCypher n'est pas rétrocompatible, car `~id` est désormais considéré comme un nom de propriété réservé. Si vous utilisez déjà `~id` en tant que propriété dans vos données et requêtes, vous devrez migrer la propriété `~id` vers une nouvelle clé de propriété avant de passer à cette version.

- Ajout de plusieurs nouveaux modes `DESCRIBE` SPARQL ainsi que d'indicateurs de requête pour les configurer.

Améliorations de cette version du moteur

- Amélioration des performances d'openCypher, en particulier pour les requêtes VLP.
- Amélioration des performances du DFE pour les requêtes Gremlin avec des limites autres que celles du terminal, telles que :

```
g.withSideEffect('Neptune#useDFE',true).V().hasLabel('Student').limit(5).out('takesCourse')
```

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.0.2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.4`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.0.2

Mise à niveau vers cette version

Amazon Neptune 1.2.0.2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.0.2.R6 du moteur Amazon Neptune (12/09/2023)

Depuis le 12 septembre 2023, la version 1.2.0.2.R6 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Note

En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :

- La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.

- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue SPARQL qui empêchait l'opérateur `REGEX` d'aboutir lorsqu'il était appelé sur un littéral balisé dans un langage.
- Résolution d'un problème qui dégradait les performances de chargement en bloc.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.0.2.R6, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.5`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.0.2.R6

Votre cluster de bases de données Neptune sera automatiquement mis à niveau vers cette version de correctif de maintenance lors de la prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.2.0.2`.

Mise à niveau vers cette version

Amazon Neptune 1.2.0.2.R6 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.0.2.R5 du moteur Amazon Neptune (16/08/2023)

Depuis le 16 août 2023, la version 1.2.0.2.R5 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Important

Les modifications apportées à cette version du moteur peuvent, dans certains cas, entraîner une dégradation des performances de chargement en bloc. Par conséquent, les mises à niveau de cette version ont été temporairement suspendues jusqu'à ce que le problème soit résolu.

Note

En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :

- La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative

de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.

- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin en raison duquel `order()` empêchait de trier correctement les sorties de chaînes lorsque certaines d'entre elles contenaient un espace.
- Correction d'un bogue Gremlin qui provoquait une fuite de transaction lors de la vérification du point de terminaison de statut d'une requête Gremlin pour détecter les requêtes contenant des prédicats dans les traversées enfants pour les étapes qui ne sont pas traitées nativement.
- Correction d'un bogue openCypher lié à la gestion des transactions Bolt.
- Correction d'un problème de simultanéité sur la couche de stockage qui pouvait provoquer un plantage.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.0.2.R5, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.5`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.0.2.R5

Votre cluster de bases de données Neptune sera automatiquement mis à niveau vers cette version de correctif de maintenance lors de la prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.2.0.2`.

Mise à niveau vers cette version

Amazon Neptune 1.2.0.2.R5 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces

instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.0.2.R4 du moteur Amazon Neptune (08/05/2023)

Depuis le 8 mai 2023, la version 1.2.0.2.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Note

En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :

- La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch

métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.

- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue SPARQL en raison duquel un grand nombre de valeurs injectées via une clause `VALUES` pouvait entraîner une dégradation des performances.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.0.2.R4, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.6`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`

- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.0.2.R4

Votre cluster de bases de données Neptune sera automatiquement mis à niveau vers cette version de correctif de maintenance lors de la prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.2.0.2`.

Mise à niveau vers cette version

Amazon Neptune 1.2.0.2.R4 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise

à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the instance before
>   proceeding with the upgrade.
> ```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.0.2.R3 du moteur Amazon Neptune (27/03/2023)

Depuis le 27 mars 2023, la version 1.2.0.2.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Note

En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :

- La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.

- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

- Pour les clusters de base de données sans serveur, le paramètre de capacité minimale a été modifié à 1,0 NCU et le paramètre maximum valide le plus bas à 2,5. NCUs Consultez Mise à l'échelle de la capacité dans un cluster de bases de données Neptune sans serveur.
- Ajout d'un `enableInterContainerTrafficEncryption` paramètre à tous les Neptune ML APIs, que vous pouvez utiliser pour activer et désactiver le chiffrement du trafic entre conteneurs dans le cadre de tâches d'entraînement ou de réglage d'hyperparamètres.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin qui empêchait la reconnaissance de la syntaxe Gremlin `option(Predicate)` comme étant valide.
- Correction d'un bogue Gremlin qui empêchait le nettoyage correct des requêtes en cas d'échec parce qu'elles contenaient trop d'étapes.

- Correction des problèmes d'exactitude Gremlin pour les requêtes DFE utilisant `limit` comme traversée enfant des étapes autres que l'union en revenant à Tinkerpop. Voici un exemple de requête de ce type :

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- Correction d'une fuite potentielle de transaction Gremlin lorsqu'une requête soumise sous forme de chaîne contient `GroupCountStep`.
- Correction d'un bogue openCypher où la valeur du type de paramètre ne faisait pas toujours l'objet d'une inférence correcte pour une liste ou une liste de mappages.
- Correction d'un bogue openCypher en raison duquel les requêtes de mise à jour et de retour ne géraient pas `orderBy`, `limit` ni `skip` correctement.
- Correction d'un bogue openCypher qui permettait de remplacer les paramètres contenus dans une requête par des paramètres contenus dans une autre requête simultanée.
- Correction d'un bogue SPARQL en raison duquel un grand nombre de valeurs injectées dans une clause `VALUES` pouvait entraîner une dégradation des performances.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.0.2.R3, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.6`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.0.2.R3

Votre cluster de bases de données Neptune sera automatiquement mis à niveau vers cette version de correctif de maintenance lors de la prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.2.0.2`.

Mise à niveau vers cette version

Amazon Neptune 1.2.0.2.R3 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.0.2.R2 du moteur Amazon Neptune (15/12/2022)

Depuis le 15 décembre 2022, la version 1.2.0.2.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :
>
> - La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
> - La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.
>
> Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.
>
> Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.
> - Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir

de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

- Amélioration des performances des requêtes openCypher impliquant `MERGE` et `OPTIONAL MATCH`.
- Amélioration des performances des requêtes openCypher impliquant `UNWIND` pour une liste de mappages de valeurs littérales.
- Amélioration des performances des requêtes openCypher dotées d'un filtre `IN` pour `id`. Par exemple :

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Amélioration des performances et corrections de divers opérateurs Gremlin, notamment `repeat`, `coalesce`, `store` et `aggregate`.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue openCypher où les requêtes renvoyaient la chaîne `"null"` au lieu d'une valeur nulle dans Bolt et SPARQL-JSON.
- Correction d'un bogue Gremlin qui empêchait la propagation d'une étiquette d'étape attachée à `UnionStep` au dernier élément de chemin de ses traversées enfants.
- Correction d'un bogue Gremlin qui empêchait l'optimisation de `valueMap()` lors d'une traversée `by()` dans le moteur DFE.
- Correction d'un bogue Gremlin qui empêchait les lignes de se verrouiller avec les requêtes de lecture exécutées dans le cadre d'une transaction Gremlin plus longue.
- Correction d'un bogue du journal d'audit qui entraînait la journalisation d'informations inutiles et l'absence de certains champs dans les journaux.
- Correction d'un bogue du journal d'audit en raison duquel l'ARN IAM des requêtes HTTP adressées à un cluster de bases de données compatible IAM n'était pas enregistré.

- Correction d'un bogue dans le cache de recherche afin de limiter la mémoire incrémentielle utilisée pour les écritures dans le cache.
- Correction d'un bogue du cache de recherche qui impliquait la définition du mode lecture seule pour le cache de recherche en cas d'échec des écritures.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.0.2.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.4`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.0.2.R2

Votre cluster de bases de données Neptune sera automatiquement mis à niveau vers cette version de correctif de maintenance lors de la prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.2.0.2`.

Mise à niveau vers cette version

Amazon Neptune 1.2.0.2.R2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que

vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.0.1 du moteur Amazon Neptune (26/10/2022)

Depuis le 26 octobre 2022, la version 1.2.0.1 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :
>
> - La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres

personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

  Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

  Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.
- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Versions de correctifs ultérieures pour cette version

- Version de maintenance : 1.2.0.1.R2 (13/12/2022)
- Version de maintenance : 1.2.0.1.R3 (27/09/2023)

Nouvelles fonctionnalités pour cette version du moteur

- Ajout d'Amazon Neptune sans serveur, une configuration d'autoscaling à la demande qui fait évoluer la capacité de votre cluster de bases de données en fonction de l'augmentation ou de la baisse des besoins de traitement.

Améliorations de cette version du moteur

- Amélioration des performances des requêtes Gremlin `order-by`. Les requêtes Gremlin avec `order-by` à la fin d'une étape `NeptuneGraphQueryStep` utilisent désormais une taille de bloc plus grande pour de meilleures performances. Cela ne s'applique pas à `order-by` sur un nœud interne (non racine) du plan de requête.
- Amélioration des performances des requêtes de mise à jour Gremlin. Les sommets et les arêtes doivent désormais être verrouillés pour empêcher toute suppression lors de l'ajout d'arêtes ou de propriétés. Cette modification élimine les verrouillages en double au sein d'une transaction, ce qui améliore les performances.
- Amélioration des performances des requêtes Gremlin qui utilisent `dedup()` l'intérieur d'une sous-requête `repeat()` en la redirigeant `dedup` jusqu'à la couche d'exécution native.
- Ajout de messages d'erreur conviviaux pour les erreurs d'authentification IAM. Ces messages affichent désormais l'ARN de votre utilisateur ou rôle IAM, l'ARN de la ressource et une liste des actions non autorisées pour la demande. La liste des actions non autorisées vous permet de voir ce qui est peut manquer ou être explicitement refusé dans la politique IAM que vous utilisez.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue de Garmlin à cause duquel l'utilisation `PartitionStrategy` après la mise à niveau vers la version TinkerPop 3.5 entraînait une erreur avec le message « PartitionStrategy ne fonctionne pas avec les traversées anonymes », qui empêchait l'exécution de la traversée.
- Correction d'un bogue d'exactitude Gremlin concernant la conversion de `WherePredicateStep`, où le moteur de requêtes de Neptune générait des résultats incorrects pour les requêtes utilisant `where(P.neq('x'))` et autres variantes.
- Correction d'un bogue openCypher dans la clause `MERGE` qui, dans certains cas, provoquait la création de nœuds et d'arêtes en double.
- Correction d'un bogue SPARQL lié au traitement des requêtes contenant `(NOT) EXISTS` dans une clause `OPTIONAL`, où il manquait les résultats des requêtes dans certains cas.

- Correction d'un bogue du chargeur en bloc qui entraînait des régressions de performance sous de fortes charges d'insertion.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.0.1, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.4`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.0.1

Mise à niveau vers cette version

Amazon Neptune 1.2.0.1 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
```

```
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.0.1.R3 du moteur Amazon Neptune (27/09/2023)

Depuis le 27 septembre 2023, la version 1.2.0.1.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :
>
> - La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles

avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

  Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

  Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.
- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

- Ajout d'un `enableInterContainerTrafficEncryption` paramètre à tous les Neptune ML APIs, que vous pouvez utiliser pour activer et désactiver le chiffrement du trafic entre conteneurs dans le cadre de tâches d'entraînement ou de réglage d'hyperparamètres.
- Pour les clusters de base de données sans serveur, le paramètre de capacité minimale a été modifié à 1,0 NCU et le paramètre maximum valide le plus bas à 2,5. NCUs Voir Mise à l'échelle de

la capacité dans un cluster de bases de données Neptune sans serveur *(((before release, this change needs to be reflected in the serverless page too)))*.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue dans OpenCypher à cause duquel les update-and-return requêtes ne se géraient pas ou `skip` ne `orderBy` fonctionnaient pas `limit` correctement.
- Correction d'un bogue openCypher qui permettait de remplacer les paramètres contenus dans une requête par des paramètres contenus dans une autre requête simultanée.
- Correction d'un bogue openCypher lié à la gestion des transactions Bolt.
- Correction des problèmes d'exactitude Gremlin pour les requêtes DFE utilisant `limit` comme traversée enfant des étapes autres que l'union en revenant à Tinkerpop. Par exemple, pour des requêtes comme celle-ci :

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
 .select("a")
 .by(out()
 .limit(1))
```

- Correction d'un bogue Gremlin qui empêchait une requête d'échouer parce qu'elle contenait trop d' TinkerPop étapes et ne pouvait donc pas être nettoyée.
- Correction d'un bogue Gremlin en raison duquel `order()` empêchait de trier correctement les sorties de chaînes lorsque certaines d'entre elles contenaient un espace.
- Correction d'un bogue Gremlin qui provoquait une fuite de transaction lorsqu'une requête était soumise sous forme de chaîne et contenait `GroupCountStep`.
- Correction d'un bogue Gremlin qui provoquait une fuite de transaction lors de la vérification du point de terminaison de statut d'une requête Gremlin pour détecter les requêtes contenant des prédicats dans les traversées enfants pour les étapes qui ne sont pas traitées nativement.
- Correction d'un bogue Gremlin où l'ajout d'une arête et de ses propriétés suivi de `inV()` ou de `outV()` levait une exception `InternalFailureException`.
- Correction d'un problème de simultanéité dans la couche de stockage.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.0.1.R3, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.6`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.0.1.R3

Votre cluster de bases de données Neptune sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur 1.2.0.1.

Mise à niveau vers cette version

Amazon Neptune 1.2.0.1.R3 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.0.1.R2 du moteur Amazon Neptune (13/12/2022)

Depuis le 13 décembre 2022, la version 1.2.0.1.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Note

En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :

- La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch

métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.

- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

- Amélioration des performances des requêtes openCypher impliquant `UNWIND` au niveau d'une liste de mappages de valeurs littérales.
- Amélioration des performances et corrections de divers opérateurs Gremlin, notamment `repeat`, `coalesce`, `store` et `aggregate`.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue openCypher où les requêtes renvoyaient la chaîne `"null"` au lieu d'une valeur nulle dans Bolt et SPARQL-JSON.
- Correction d'un bogue du journal d'audit qui entraînait la journalisation d'informations inutiles et l'absence de certains champs dans les journaux.

- Correction d'un bogue dans le cache de recherche afin de limiter la mémoire incrémentielle utilisée pour les écritures dans le cache.
- Correction d'un bogue du cache de recherche qui impliquait la définition du mode lecture seule pour le cache de recherche en cas d'échec des écritures.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.0.1.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.4`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.0.1.R2

Votre cluster de bases de données Neptune sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur 1.2.0.1.

Mise à niveau vers cette version

Amazon Neptune 1.2.0.1.R2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un

instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.0.0 du moteur Amazon Neptune (21/07/2022)

Depuis le 21 juillet 2022, la version 1.2.0.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :
>
> - La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de

la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau depuis une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

  Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

  Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.
- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Versions de correctifs ultérieures pour cette version

- Sortie : 1.2.0.0.R2 (14/10/2022)
- Sortie : 1.2.0.0.R3 (15/12/2022)
- Sortie : 1.2.0.0.R4 (29/09/2023)

Nouvelles fonctionnalités pour cette version du moteur

- Ajout de la prise en charge des bases de données globales. Une base de données globale Neptune couvre plusieurs Régions AWS bases de données et se compose d'un cluster de bases de données principal dans une région et d'un maximum de cinq clusters de bases de données secondaires dans d'autres régions.
- Ajout de la prise en charge d'un contrôle d'accès plus granulaire dans les politiques IAM Neptune que ce qui était disponible auparavant, sur la base des actions du plan de données. Il s'agit d'un changement radical dans la mesure où les politiques IAM existantes basées sur l'action `connect` obsolète doivent être ajustées pour utiliser les actions du plan de données plus granulaires. Consultez Types de politique IAM.
- Disponibilité améliorée des instances de lecteur. Auparavant, lors du redémarrage d'une instance d'enregistreur, toutes les instances de lecteur d'un cluster Neptune redémarraient également. À partir de la version 1.2.0.0 du moteur, les instances de lecteur restent actives après le redémarrage de l'instance d'enregistreur, ce qui améliore la disponibilité des instances de lecteur. Les instances de lecteur peuvent être redémarrées séparément pour prendre en compte les modifications apportées aux groupes de paramètres. Consultez Redémarrage d'une instance de base de données dans Amazon Neptune.
- Ajout d'un nouveau paramètre de cluster de bases de données neptune_streams_expiry_days qui vous permet de définir le nombre de jours pendant lesquels les enregistrements de flux sont conservés sur le serveur avant d'être supprimés. Cette plage est comprise entre 1 et 90, et la valeur par défaut est 7.

Améliorations de cette version du moteur

- Performances de sérialisation Gremlin améliorées pour ByteCode les requêtes.
- Neptune traite désormais les prédicats de texte à l'aide du moteur DFE, pour de meilleures performances.
- Neptune traite désormais les étapes `limit()` Gremlin à l'aide du moteur DFE, y compris les limites de traversée hors terminal et enfant.
- Modification de la gestion DFE de l'étape `union()` Gremlin pour qu'elle fonctionne avec d'autres nouvelles fonctionnalités, ce qui signifie que les nœuds de référence apparaissent dans les profils de requête comme prévu.
- Performances améliorées jusqu'à cinq fois de certaines opérations de jointure coûteuses au sein du DFE en les parallélisant.

- Ajout de la prise en charge de la modulation `by()` pour `OrderGlobalStep order(global)` pour le moteur DFE Gremlin.
- Ajout de l'affichage des valeurs statiques injectées dans les détails de la fonctionnalité explain du DFE.
- Amélioration des performances lors du nettoyage des modèles dupliqués.
- Ajout de la prise en charge de la préservation des commandes dans le moteur DFE Gremlin.
- Amélioration des performances des requêtes Gremlin comportant des filtres vides, par exemple :

```
g.V().hasId(P.within([]))
```

```
g.V().hasId([])
```

- Amélioration des messages d'erreur lorsqu'une requête SPARQL utilise une valeur numérique trop grande pour que Neptune puisse la représenter en interne.
- Amélioration des performances de la suppression de sommets associés à des arêtes en réduisant les recherches d'index lorsque les flux sont désactivés.
- Support DFE étendu à un plus grand nombre de variantes de l'`has()`étape, en particulier à `hasKey()``hasLabel()`, et à la plage de prédicats pour strings/URIs within. `has()` Cela concerne les requêtes telles que les suivantes :

```
// hasKey() on properties
g.V().properties().hasKey("name")
g.V().properties().has(T.key, TextP.startingWith("a"))
g.E().properties().hasKey("weight")
g.E().properties().hasKey(TextP.containing("t"))

// hasLabel() on vertex properties
g.V().properties().hasLabel("name")

// range predicates on ID and Label fields
g.V().has(T.label, gt("person"))
g.E().has(T.id, lte("(an ID value)"))
```

- Ajout d'une fonction openCypher `join()` spécifique à Neptune qui concatène les chaînes d'une liste en une seule chaîne.
- Mise à jour des politiques gérées par Neptune pour inclure les autorisations d'accès aux données et les autorisations pour la nouvelle base de données mondiale. APIs

Défauts corrigés dans cette version du moteur

- Correction d'un bogue en raison duquel une requête HTTP sans type de contenu spécifié échouait automatiquement.
- Correction d'un bogue SPARQL dans l'optimiseur de requêtes qui empêchait l'utilisation d'un appel de service dans une requête.
- Correction d'un bogue SPARQL dans l'analyseur Turtle RDF à cause duquel une combinaison particulière de données Unicode provoquait un échec.
- Correction d'un bogue SPARQL en raison duquel une combinaison particulière de clauses `GRAPH` et `SELECT` générait des résultats de requête incorrects.
- Correction d'un bogue Gremlin qui provoquait un problème d'exactitude pour les requêtes utilisant n'importe quelle étape de filtre au sein d'une étape d'union, comme suit :

```
g.V("1").union(hasLabel("person"), out())
```

- Correction d'un bogue Gremlin où la valeur `count()` de `both().simplePath()` multipliait par deux le nombre réel de résultats renvoyés sans `count()`.
- Correction d'un bogue openCypher en raison duquel une exception de non-correspondance de signature défectueuse était générée par le serveur pour les requêtes Bolt adressées aux clusters avec l'authentification IAM activée.
- Correction d'un bogue openCypher où une requête utilisant HTTP keep-alive pouvait être fermée de manière incorrecte si elle était soumise après l'échec d'une demande.
- Correction d'un bogue openCypher qui pouvait provoquer le déclenchement d'une erreur interne lorsqu'une requête renvoyant une valeur de constante était soumise.
- Correction d'un bogue lié aux détails de la fonctionnalité explaon, de sorte que la sous-requête DFE `Time(ms)` additionne désormais correctement les temps CPU des opérateurs au sein de la sous-requête DFE. Prenons l'extrait suivant représentant la sortie de la fonctionnalité explain à titre d'exemple :

```
subQuery1
#######################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments                            #
 Mode # Units In # Units Out # Ratio # Time (ms) #
#######################################################################################
  ...
#######################################################################################
```

```
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=...graph#336e.../graph_1 #
 -    # 1        # 1         # 1.00  # 0.38      #
#    #        #        #                       # coordinationTime(ms)=0.026        #
     #          #           #       #           #
##########################################################################################
  ...
subQuery=...graph#336e.../graph_1
##########################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments                          #
 Mode # Units In # Units Out # Ratio # Time (ms) #
##########################################################################################
# 0  # 1      # -      # DFESolutionInjection  # solutions=[?100 -> [-10^^<LONG>]] #
 -    # 0        # 1         # 0.00  # 0.04      #
#    #        #        #                       # outSchema=[?100]                   #
     #          #           #       #           #
##########################################################################################
# 1  # 3      # -      # DFERelationalJoin     # joinVars=[]                        #
 -    # 2        # 1         # 0.50  # 0.29      #
##########################################################################################
# 2  # 1      # -      # DFESolutionInjection  # outSchema=[]                      #
 -    # 0        # 1         # 0.00  # 0.01      #
##########################################################################################
# 3  # -      # -      # DFEDrain              # -                                  #
 -    # 1        # 0         # 0.00  # 0.02      #
##########################################################################################
```

Le total des durées des sous-requêtes dans la dernière colonne du tableau inférieur s'élève à 0,36 ms (`.04 + .29 + .01 + .02 = .36`). Lorsque vous ajoutez le temps de coordination de cette sous-requête (`.36 + .026 = .386`), vous obtenez un résultat proche de a durée de la sous-requête enregistrée dans la dernière colonne du tableau supérieur, à savoir `0.38` ms.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.0.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.4`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.0.0

Comme il s'agit d'une version majeure du moteur, la mise à niveau n'est pas automatique.

Vous ne pouvez effectuer une mise à niveau pour publier `1.2.0.0` manuellement qu'à partir de la dernière version de correctif du moteur 1.1.1.0. Les versions antérieures du moteur doivent d'abord être mises à niveau vers la dernière version `1.1.1.0` avant de pouvoir passer à la version `1.2.0.0`.

Par conséquent, avant d'essayer de passer à cette version, veuillez vérifier que vous exécuté actuellement le dernier correctif de la version `1.1.1.0`. Si ce n'est pas le cas, commencez par effectuer une mise à niveau vers la dernière version du correctif de `1.1.1.0`.

Avant la mise à niveau, vous devez également recréer tout groupe de paramètres de cluster de bases de données personnalisé que vous utilisiez avec votre version précédente, à l'aide de la famille `neptune1.2` de groupes de paramètres. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

Si vous effectuez d'abord une mise à niveau vers une version, `1.1.1.0` puis immédiatement vers une version `1.2.0.0`, vous risquez de rencontrer une erreur telle que la suivante :

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer (voir Maintenance du cluster de bases de données Amazon Neptune).

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations,

l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.0.0.R4 du moteur Amazon Neptune (29/09/2023)

Depuis le 29 septembre 2023, la version 1.2.0.0.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :
>
> - La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
> - La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.
>
> Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.

- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

- Ajout d'un `enableInterContainerTrafficEncryption` paramètre à tous les Neptune ML APIs, que vous pouvez utiliser pour activer et désactiver le chiffrement du trafic entre conteneurs dans le cadre de tâches d'entraînement ou de réglage d'hyperparamètres.
- Pour les clusters de base de données sans serveur, le paramètre de capacité minimale a été modifié à 1,0 NCU et le paramètre maximum valide le plus bas à 2,5. NCUs Voir Mise à l'échelle de la capacité dans un cluster de bases de données Neptune sans serveur *(((before release, this change needs to be reflected in the serverless page too)))*.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue dans OpenCypher à cause duquel les update-and-return requêtes ne se géraient pas ou `skip` ne `orderBy` fonctionnaient pas `limit` correctement.
- Correction d'un bogue openCypher qui permettait de remplacer les paramètres contenus dans une requête par des paramètres contenus dans une autre requête simultanée.
- Correction d'un bogue openCypher lié à la gestion des transactions Bolt.
- Correction des problèmes d'exactitude Gremlin pour les requêtes DFE utilisant `limit` comme traversée enfant des étapes autres que l'union en revenant à Tinkerpop. Par exemple, pour des requêtes comme celle-ci :

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
```

```
  .select("a")
  .by(out()
  .limit(1))
```

- Correction d'un bogue Gremlin qui empêchait une requête d'échouer parce qu'elle contenait trop d' TinkerPop étapes et ne pouvait donc pas être nettoyée.
- Correction d'un bogue Gremlin en raison duquel `order()` empêchait de trier correctement les sorties de chaînes lorsque certaines d'entre elles contenaient un espace.
- Correction d'un bogue Gremlin qui provoquait une fuite de transaction lorsqu'une requête était soumise sous forme de chaîne et contenait `GroupCountStep`.
- Correction d'un bogue Gremlin qui provoquait une fuite de transaction lors de la vérification du point de terminaison de statut d'une requête Gremlin pour détecter les requêtes contenant des prédicats dans les traversées enfants pour les étapes qui ne sont pas traitées nativement.
- Correction d'un bogue Gremlin où l'ajout d'une arête et de ses propriétés suivi de `inV()` ou de `outV()` levait une exception `InternalFailureException`.
- Correction d'un problème de simultanéité dans la couche de stockage.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.0.0.R4, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.6`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.0.0.R4

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.2.0.0`.

Vous ne pouvez effectuer une mise à niveau pour publier `1.2.0.0` manuellement qu'à partir de la dernière version de correctif du moteur 1.1.1.0. Les versions antérieures du moteur doivent d'abord être mises à niveau vers la dernière version `1.1.1.0` avant de pouvoir passer à la version `1.2.0.0`.

Si vous effectuez d'abord une mise à niveau vers une version, `1.1.1.0` puis immédiatement vers une version `1.2.0.0`, vous risquez de rencontrer une erreur telle que la suivante :

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
--db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que

vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.0.0.R3 du moteur Amazon Neptune (15/12/2022)

Depuis le 15 décembre 2022, la version 1.2.0.0.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :
>
> - La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres

personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.

- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau depuis une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

  Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

  Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.
- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

- Amélioration des performances des requêtes openCypher impliquant `MERGE` et `OPTIONAL MATCH`.
- Amélioration des performances des requêtes openCypher impliquant `UNWIND` au niveau d'une liste de mappages de valeurs littérales.

- Amélioration des performances des requêtes openCypher dotées d'un filtre `IN` pour `id`. Par exemple :

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Amélioration des performances et corrections de divers opérateurs Gremlin, notamment `repeat`, `coalesce`, `store` et `aggregate`.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue openCypher où les requêtes renvoyaient la chaîne `"null"` au lieu d'une valeur nulle dans Bolt et SPARQL-JSON.
- Correction d'un bogue openCypher afin de pouvoir interpréter correctement le type de paramètre lorsque la valeur est une liste ou une liste de mappages.
- Correction d'un bogue du journal d'audit qui entraînait la journalisation d'informations inutiles et l'absence de certains champs dans les journaux.
- Correction d'un bogue du journal d'audit en raison duquel l'ARN IAM des requêtes HTTP adressées à un cluster de bases de données compatible IAM n'était pas enregistré.
- Correction d'un bogue dans le cache de recherche afin de limiter la mémoire incrémentielle utilisée pour les écritures dans le cache.
- Correction d'un bogue du cache de recherche qui impliquait la définition du mode lecture seule pour le cache de recherche en cas d'échec des écritures.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.0.0.R3, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.4`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.0.0.R3

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.2.0.0`.

Vous ne pouvez effectuer une mise à niveau pour publier `1.2.0.0` manuellement qu'à partir de la dernière version de correctif du moteur 1.1.1.0. Les versions antérieures du moteur doivent d'abord être mises à niveau vers la dernière version `1.1.1.0` avant de pouvoir passer à la version `1.2.0.0`.

Si vous effectuez d'abord une mise à niveau vers une version, `1.1.1.0` puis immédiatement vers une version `1.2.0.0`, vous risquez de rencontrer une erreur telle que la suivante :

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
```

```
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.2.0.0.R2 du moteur Amazon Neptune (14/10/2022)

Depuis le 14 octobre 2022, la version 1.2.0.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Note**
>
> En cas de mise à niveau à partir d'une version de moteur antérieure à 1.2.0.0 :

- La version 1.2.0.0 du moteur implique un nouveau format pour les groupes de paramètres personnalisés et les groupes de paramètres de cluster personnalisés. Par conséquent, si vous effectuez une mise à niveau d'une version de moteur antérieure vers la version 1.2.0.0 ou une version supérieure, vous devrez recréer tous vos groupes de paramètres personnalisés et groupes de paramètres de cluster personnalisés existants à l'aide de la famille de groupes de paramètres `neptune1.2`. Les versions antérieures utilisaient une famille de groupes de paramètres `neptune1`, lesquels ne sont pas compatibles avec les versions 1.2.0.0 et supérieures. Pour plus d'informations, consultez Groupes de paramètres Amazon Neptune.
- La version 1.2.0.0 du moteur comprend également un nouveau format pour les journaux d'annulation. Par conséquent, tous les journaux d'annulation créés par une version antérieure du moteur doivent être purgés et la `UndoLogsListSize` CloudWatch métrique doit tomber à zéro avant qu'une mise à niveau à partir d'une version antérieure à 1.2.0.0 puisse commencer. S'il existe trop d'enregistrements de journaux d'annulation (200 000 entrées ou plus) lorsque vous essayez de démarrer une mise à jour, la tentative de mise à niveau peut expirer en attendant que la purge des journaux d'annulation soit terminée.

  Vous pouvez accélérer le taux de purge en mettant à niveau l'instance d'enregistreur du cluster, où la purge a lieu. Suivre cette étape avant d'essayer de procéder à la mise à niveau contribue à réduire le nombre de journaux d'annulation avant de commencer. L'augmentation de la taille du dispositif d'écriture dans un type d'instance 24XL peut accroître le taux de purge, permettant ainsi de traiter plus d'un million d'enregistrements par heure.

  Si l'`UndoLogsListSize` CloudWatch indicateur est extrêmement important, l'ouverture d'un dossier de support peut vous aider à explorer d'autres stratégies pour le réduire.
- Enfin, une modification majeure a été apportée à la version 1.2.0.0. Celle-ci concerne le code antérieur qui utilisait le protocole Bolt avec l'authentification IAM. À partir de la version 1.2.0.0, Bolt a besoin d'un chemin de ressources pour la signature IAM. En Java, la définition du chemin de ressources peut ressembler à ceci : `request.setResourcePath("/openCypher"));`. Dans d'autres langages, `/openCypher` peut être ajouté à l'URI du point de terminaison. Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

- Amélioration des performances des requêtes Gremlin `order-by`. Les requêtes Gremlin avec `order-by` à la fin d'une étape `NeptuneGraphQueryStep` utilisent désormais une taille de bloc plus grande pour de meilleures performances. Cela ne s'applique pas à `order-by` sur un nœud interne (non racine) du plan de requête.
- Amélioration des performances des requêtes de mise à jour Gremlin. Les sommets et les arêtes doivent désormais être verrouillés pour empêcher toute suppression lors de l'ajout d'arêtes ou de propriétés. Cette modification élimine les verrouillages en double au sein d'une transaction, ce qui améliore les performances.
- Amélioration des performances des requêtes Gremlin qui utilisent `dedup()` l'intérieur d'une sous-requête `repeat()` en la redirigeant `dedup` jusqu'à la couche d'exécution native.
- Ajout de l'indicateur de requête Gremlin `Neptune#cardinalityEstimates`. Lorsque cette valeur est définie sur `false`, les estimations de cardinalité sont désactivées.
- Ajout de messages d'erreur conviviaux pour les erreurs d'authentification IAM. Ces messages affichent désormais l'ARN de votre utilisateur ou rôle IAM, l'ARN de la ressource et une liste des actions non autorisées pour la demande. La liste des actions non autorisées vous permet de voir ce qui est peut manquer ou être explicitement refusé dans la politique IAM que vous utilisez.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue d'exactitude Gremlin concernant la conversion de `WherePredicateStep`, où le moteur de requêtes de Neptune générait des résultats incorrects pour les requêtes utilisant `where(P.neq('x'))` et autres variantes.
- Correction d'un bogue de Garmlin à cause duquel l'utilisation `PartitionStrategy` après la mise à niveau vers la version TinkerPop 3.5 entraînait une erreur avec le message « PartitionStrategy ne fonctionne pas avec les traversées anonymes », qui empêchait l'exécution de la traversée.
- Correction de divers bogues Gremlin liés à la valeur `joinTime` d'une jointure finale et aux statistiques au sein des sous-groupes `Project.ASK`.
- Correction d'un bogue openCypher dans la clause `MERGE` qui, dans certains cas, provoquait la création de nœuds et d'arêtes en double.
- Correction d'un bogue de transaction qui permettait à une session d'insérer des données de graphe et de les valider même lorsque les insertions de dictionnaire simultanées correspondantes étaient annulées.

- Correction d'un bogue du chargeur en bloc qui entraînait des régressions de performance sous de fortes charges d'insertion.
- Correction d'un bogue SPARQL lié au traitement des requêtes contenant `(NOT) EXISTS` dans une clause `OPTIONAL`, où il manquait les résultats des requêtes dans certains cas.
- Correction d'un bogue à cause duquel les pilotes pouvaient sembler bloqués lorsque les demandes étaient annulées en raison de leur expiration avant le début de leur évaluation. Il était possible de parvenir à cet état si tous les threads de traitement des requêtes du serveur étaient consommés pendant que des éléments de la file d'attente des demandes arrivaient à expiration. Comme l'expiration d'élément dans la file d'attente des demandes n'entraînait pas l'envoi immédiat de messages, les réponses semblaient rester en attente pour le client.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.2.0.0.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.4`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.2.0.0.R2

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.2.0.0`.

Vous ne pouvez effectuer une mise à niveau pour publier `1.2.0.0` manuellement qu'à partir de la dernière version de correctif du moteur 1.1.1.0. Les versions antérieures du moteur doivent d'abord être mises à niveau vers la dernière version `1.1.1.0` avant de pouvoir passer à la version `1.2.0.0`.

Si vous effectuez d'abord une mise à niveau vers une version, `1.1.1.0` puis immédiatement vers une version `1.2.0.0`, vous risquez de rencontrer une erreur telle que la suivante :

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
```

```
    proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.1.0.0 du moteur Amazon Neptune (19/04/2022)

Depuis le 19 avril 2022, la version 1.1.1.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau. Afin de mener à bien la mise à niveau, chaque sous-réseau de chaque zone de disponibilité (AZ) doit disposer d'au moins une adresse IP disponible par instance Neptune. Par exemple, s'il existe une instance d'enregistreur et deux instances de lecteur dans le sous-réseau 1, et deux instances de lecteur dans le sous-réseau 2, le sous-réseau 1 doit avoir au moins trois adresses IP libres, tandis que le sous-réseau 2 doit avoir au moins deux adresses IP libres avant de commencer la mise à niveau.

Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera alors indisponible pendant quelques minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Versions de correctifs ultérieures pour cette version

- Version de maintenance 1.1.1.0.R2 (16/05/2022)
- Sortie : 1.1.1.0.R3 (07/06/2022)
- Sortie : 1.1.1.0.R4 (23/06/2022)
- Sortie : 1.1.1.0.R5 (21/07/2022)
- Sortie : 1.1.1.0.R6 (23/09/2022)
- Sortie : 1.1.1.0.R7 (23/01/2023)

Nouvelles fonctionnalités pour cette version du moteur

- Le langage de requête openCypher est maintenant disponible de manière globale en production.

  > **Warning**
  >
  > Cette version apporte une modification majeure du code qui utilise openCypher avec l'authentification IAM. Dans la version préliminaire de Neptune pour openCypher, la chaîne hôte de la signature IAM incluait le protocole, tel que `bolt://`. Par exemple :
  >
  > ```
  > "Host":"bolt://(host URL):(port)"
  > ```
  >
  > À compter de cette version du moteur, ce protocole doit être omis :
  >
  > ```
  > "Host":"(host URL):(port)"
  > ```
  >
  > Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

- Ajout du support pour TinkerPop `3.5.2`. Parmi les modifications apportées dans cette version figurent la prise en charge des transactions à distance et la prise en charge du bytecode pour les sessions (avec `g.tx`), ainsi que l'ajout de la fonction `datetime()` au langage Gremlin.

  > **Warning**
  >
  > Plusieurs modifications importantes introduites dans les TinkerPop versions 3.5.0, 3.5.1 et 3.5.2 peuvent affecter votre code Gremlin. Par exemple, l'utilisation de traversées générées par un enfant GraphTraversalSource comme celle-ci ne fonctionnera plus :. `g.V().union(identity(), g.V())`
  > Maintenant, utilisez plutôt une traversée anonyme comme celle-ci : `g.V().union(identity(), __.V())`.

- Ajout de la prise en charge des clés de condition globales AWS que vous pouvez utiliser dans les stratégies d'accès aux données IAM qui contrôlent l'accès aux données stockées dans un cluster de bases de données Neptune.
- Le moteur de requêtes Neptune DFE est désormais disponible globalement pour une utilisation en production avec le langage de requête openCypher, mais pas encore pour les requêtes

Gremlin et SPARQL. Vous l'activez désormais en utilisant son propre paramètre d'instance neptune_dfe_query_engine plutôt que le paramètre lab-mode.

Améliorations de cette version du moteur

- Ajout de nouvelles fonctionnalités à openCypher, telles que la prise en charge des requêtes paramétrées, la mise en cache de l'arbre syntaxique abstrait (AST) pour les requêtes paramétrées, des améliorations des chemins de longueur variable (VLP) ainsi que de nouveaux opérateurs et clauses. Consultez Conformité aux spécifications OpenCypher dans Amazon Neptune pour découvrir le niveau actuel de prise en charge des différents langages.
- Amélioration significative des performances d'openCypher pour les charges de travail de lecture et d'écriture simples, ce qui se traduit par un débit supérieur à celui de la version 1.1.0.0.
- Suppression des limitations bidirectionnelles et des limitations de profondeur d'openCypher gérant les chemins de longueur variable.
- Prise en charge complèle dans le moteur DFE des prédicats Gremlin `within` et `without`, y compris lorsqu'ils sont combinés avec d'autres opérateurs de prédicats. Par exemple :

```
g.V().has('age', within(12, 15, 18).or(gt(30)))
```

- Prise en charge étendue dans le moteur DFE de l'étape Gremlin `order` lorsque la portée est globale (c'est-à-dire autre qu'`order(local)`) et lorsque les modulateurs `by()` ne sont pas utilisés. Par exemple, cette requête bénéficierait désormais de la prise en charge DFE :

```
g.V().values("age").order()
```

- Ajout d'un champ `isLastOp` au format de réponse du journal des modifications de flux Neptune pour indiquer qu'un enregistrement est la dernière opération de sa transaction.
- Amélioration significative des performances de journalisation des audits et réduction de la latence lorsque la journalisation des audits est activée.
- Conversion du WebSocket bytecode Garmlin et des requêtes HTTP dans un format lisible par l'utilisateur dans les journaux d'audit. Les requêtes peuvent désormais être directement copiées depuis les journaux d'audit afin de pouvoir être exécutées dans les blocs-notes Neptune et ailleurs. Notez que cette modification du format actuel du journal d'audit constitue une modification majeure.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin rare en raison duquel aucun résultat n'était renvoyé lors de l'utilisation combinée des étapes `filter()` et `count()` imbriquées, comme dans la requête suivante :

```
g.V("1").filter(out("knows")
        .filter(in("knows")
        .hasId("notExists"))
        .count())
```

- Correction d'un bogue Gremlin qui renvoyait une erreur lors de l'utilisation d'un sommet stocké par une étape agrégée dans les traversées `to()` ou `from()` associées à une étape `addE`. Voici un exemple de requête de ce type :

```
g.V("id").aggregate("v").out().addE().to(select("v").unfold()))
```

- Correction d'un bogue Gremlin en raison duquel l'étape `not` échouait dans le cas des arêtes lors de l'utilisation du moteur DFE. Par exemple :

```
g.V().not(V())
```

- Correction d'un bogue Gremlin qui entraînait l'indisponibilité des valeurs `sideEffect` dans les traversées `to()` et `from()`.
- Correction d'un bogue qui provoquait parfois une réinitialisation rapide déclenchant un basculement d'instance.
- Correction d'un bogue lié au chargeur en bloc, qui empêchait la fermeture d'une transaction ayant échoué avant le début de la prochaine tâche de chargement.
- Correction d'un bogue lié au chargeur en bloc, en raison duquel une insuffisance de mémoire pouvait provoquer un incident au niveau du système.
- Ajout d'une nouvelle tentative pour corriger un bogue lié au chargeur en bloc, à cause duquel le chargeur n'attendait pas assez longtemps que les informations d'identification IAM soient disponibles après un basculement.
- Correction d'un bogue en raison duquel le cache d'informations d'identification interne n'était pas correctement vidé pour les points de terminaison autres que ceux des requêtes, tels que le point de terminaison `status`.

- Correction d'un bogue lié aux flux pour garantir que les numéros de séquence de validation des flux sont correctement ordonnés.
- Correction d'un bogue en raison duquel les connexions de longue durée étaient interrompues avant dix jours sur les clusters compatibles IAM.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.1.0.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.4`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.1.1.0

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version. Notez que les versions antérieures à 1.1.0.0 mettront plus de temps à passer à cette version de moteur majeure.

La mise à niveau vers cette version n'est pas automatique.

Mise à niveau vers cette version

> **Important**
>
> La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau. Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez

l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera indisponible à ce stade pendant environ six minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine neptune \
    --engine-version 1.1.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine neptune ^
    --engine-version 1.1.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser une mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.1.1.0.R7 du moteur Amazon Neptune (23/01/2023)

Depuis le 23 janvier 2023, la version 1.1.1.0.R7 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
Afin de mener à bien la mise à niveau, chaque sous-réseau de chaque zone de disponibilité (AZ) doit disposer d'au moins une adresse IP disponible par instance Neptune. Par exemple, s'il existe une instance d'enregistreur et deux instances de lecteur dans le sous-réseau 1, et deux instances de lecteur dans le sous-réseau 2, le sous-réseau 1 doit avoir au moins trois adresses IP libres, tandis que le sous-réseau 2 doit avoir au moins deux adresses IP libres avant de commencer la mise à niveau.
Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera alors indisponible pendant quelques minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :

- `Applying off-line patches to DB instance`
- `DB instance shutdown`
- `Finished applying off-line patches to DB instance`
- `DB instance restarted`

Améliorations de cette version du moteur

- Amélioration des performances des requêtes openCypher impliquant `MERGE` et `OPTIONAL MATCH`.
- Amélioration des performances des requêtes openCypher impliquant `UNWIND` pour une liste de mappages de valeurs littérales.
- Amélioration des performances des requêtes openCypher dotées d'un filtre `IN` pour `id`. Par exemple :

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Amélioration des performances et corrections de divers opérateurs Gremlin, notamment `repeat`, `coalesce`, `store` et `aggregate`.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue openCypher où une demande utilisant HTTP keep-alive pouvait être fermée de manière incorrecte si elle était soumise après l'échec d'une demande.
- Correction d'un bogue openCypher où le type de paramètre n'était pas toujours correctement interprété pour une liste ou une liste de mappages.
- Correction d'un bogue openCypher où les requêtes renvoyaient la chaîne `"null"` au lieu d'une valeur nulle dans Bolt et SPARQL-JSON.
- Codes d'erreur et messages d'erreur OpenCypher corrigés en cas d'échec ou d'erreur liés au délai d'expiration des requêtes. out-of-memory
- Correction d'un bogue Gremlin qui empêchait l'optimisation de `valueMap()` lors d'une traversée `by()` dans le moteur DFE.
- Correction d'un problème lié à la logique du détecteur de blocage qui empêchait parfois le moteur de répondre.
- Correction d'un bogue du journal d'audit qui entraînait la journalisation d'informations inutiles et l'absence de certains champs dans les journaux.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.1.1.0.R7, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.3`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.1.1.0.R7

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.1.1.0`.

Mise à niveau vers cette version

Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera indisponible à ce stade pendant environ six minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
> ```

```
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.1.1.0.R6 du moteur Amazon Neptune (23/09/2022)

Depuis le 23 septembre 2022, la version 1.1.1.0.R6 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
Afin de mener à bien la mise à niveau, chaque sous-réseau de chaque zone de disponibilité (AZ) doit disposer d'au moins une adresse IP disponible par instance Neptune. Par exemple, s'il existe une instance d'enregistreur et deux instances de lecteur dans le sous-réseau 1, et deux instances de lecteur dans le sous-réseau 2, le sous-réseau 1 doit avoir au moins trois adresses IP libres, tandis que le sous-réseau 2 doit avoir au moins deux adresses IP libres avant de commencer la mise à niveau.
Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.

Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera alors indisponible pendant quelques minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Cette version apporte une modification majeure du code qui utilise openCypher avec l'authentification IAM. Jusqu'à présent, la chaîne d'hôte contenue dans la signature IAM incluait le protocole `bolt://`, par exemple :

```
"Host":"bolt://(host URL):(port)"
```

À compter de la version du moteur `1.1.1.0`, ce protocole doit être omis :

```
"Host":"(host URL):(port)"
```

Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

- Amélioration des performances des requêtes Gremlin `order-by`. Les requêtes Gremlin avec `order-by` à la fin d'une étape `NeptuneGraphQueryStep` utilisent désormais une taille de bloc plus grande pour de meilleures performances. Cela ne s'applique pas à `order-by` sur un nœud interne (non racine) du plan de requête.
- Amélioration des performances des requêtes de mise à jour Gremlin. Les sommets et les arêtes doivent désormais être verrouillés pour empêcher toute suppression lors de l'ajout d'arêtes ou de propriétés. Cette modification élimine les verrouillages en double au sein d'une transaction, ce qui améliore les performances.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue openCypher dans la clause `MERGE` qui, dans certains cas, provoquait la création de nœuds et d'arêtes en double.
- Correction d'un bogue lié au traitement des requêtes SPARQL contenant `(NOT) EXISTS` dans une clause `OPTIONAL` où il manquait les résultats des requêtes dans certains cas.
- Correction d'un bogue qui retardait le redémarrage du serveur lorsqu'un chargement en bloc était en cours.
- Correction d'un bogue où la traversée bidirectionnelle d'un modèle de longueur variable openCypher avec un filtre sur la propriété de relation provoquait une erreur. Un exemple de modèle à longueur variable de ce type est `(n)-[r*1..2]->(m)`.
- Correction d'un bogue lié à la manière dont les données mises en cache sont renvoyées au client, avec une latence étonnamment longue dans certains cas.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.1.1.0.R6, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.4`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.1.1.0.R6

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.1.1.0`.

Mise à niveau vers cette version

Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera indisponible à ce stade pendant environ six minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`

```
    • DB instance restarted
```

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations,

l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.1.1.0.R5 du moteur Amazon Neptune (21/07/2022)

Depuis le 21 juillet 2022, la version 1.1.1.0.R5 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
Afin de mener à bien la mise à niveau, chaque sous-réseau de chaque zone de disponibilité (AZ) doit disposer d'au moins une adresse IP disponible par instance Neptune. Par exemple, s'il existe une instance d'enregistreur et deux instances de lecteur dans le sous-réseau 1, et deux instances de lecteur dans le sous-réseau 2, le sous-réseau 1 doit avoir au moins trois adresses IP libres, tandis que le sous-réseau 2 doit avoir au moins deux adresses IP libres avant de commencer la mise à niveau.
Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera alors indisponible pendant quelques minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`

    - `Database cluster major version has been upgraded`
  - Messages d'événements par instance :
    - `Applying off-line patches to DB instance`
    - `DB instance shutdown`
    - `Finished applying off-line patches to DB instance`
    - `DB instance restarted`

> **Note**
>
> Cette version apporte une modification majeure du code qui utilise openCypher avec l'authentification IAM. Jusqu'à présent, la chaîne d'hôte contenue dans la signature IAM incluait le protocole `bolt://`, par exemple :
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> À compter de la version du moteur `1.1.1.0`, ce protocole doit être omis :
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

- Améliorations apportées pour prendre en charge la détection des blocages.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue qui empêchait l'arrêt complet des clusters de bases de données dans certaines conditions.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.1.1.0.R5, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.4`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.1.1.0.R5

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.1.1.0`.

Mise à niveau vers cette version

Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera indisponible à ce stade pendant environ six minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`

- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.1.1.0.R4 du moteur Amazon Neptune (23/06/2022)

Depuis le 23 juin 2022, la version 1.1.1.0.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

⚠ Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
Afin de mener à bien la mise à niveau, chaque sous-réseau de chaque zone de disponibilité (AZ) doit disposer d'au moins une adresse IP disponible par instance Neptune. Par exemple, s'il existe une instance d'enregistreur et deux instances de lecteur dans le sous-réseau 1, et deux instances de lecteur dans le sous-réseau 2, le sous-réseau 1 doit avoir au moins trois adresses IP libres, tandis que le sous-réseau 2 doit avoir au moins deux adresses IP libres avant de commencer la mise à niveau.
Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera alors indisponible pendant quelques minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :

    - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
    - `Database cluster major version has been upgraded`
  - Messages d'événements par instance :
    - `Applying off-line patches to DB instance`
    - `DB instance shutdown`
    - `Finished applying off-line patches to DB instance`
    - `DB instance restarted`

> **Note**
>
> Cette version apporte une modification majeure du code qui utilise openCypher avec l'authentification IAM. Jusqu'à présent, la chaîne d'hôte contenue dans la signature IAM incluait le protocole `bolt://`, par exemple :
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> À compter de la version du moteur `1.1.1.0`, ce protocole doit être omis :
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

- Configuration d'instance mise à jour pour les types d'instance `x2g`.
- Amélioration des performances des chutes de sommets.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin qui empêchait les solutions de maintenir un ordre stable pour une requête appelée plusieurs fois ou sur plusieurs lecteurs pour certains types de jointures ASK.

- Nous avons également réduit la portée d'une modification apportée à la version précédente qui entraînait des régressions de performance pour certains types de jointures ASK dans Gremlin.
- Correction d'un bogue Gremlin dans l'étape `union()` qui se produisait en cas d'entrée d'arête et de traversée vers un sommet dans les traversées enfants.
- Correction d'un bogue dans le profil Gremlin qui signalait que certaines étapes n'étaient pas optimisées alors qu'elles l'étaient.
- Correction d'un bogue SPARQL où les variables utilisées dans des expressions `FILTER` imbriquées dans des clauses `UNION` se voyaient attribuer des informations de portée non valides.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.1.1.0.R4, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.4`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.1.1.0.R4

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.1.1.0`.

Mise à niveau vers cette version

> **Important**
>
> La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau. Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de

votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera indisponible à ce stade pendant environ six minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un

instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.1.1.0.R3 du moteur Amazon Neptune (07/06/2022)

Depuis le 7 juin 2022, la version 1.1.1.0.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Important**
>
> La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.

Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera alors indisponible pendant quelques minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Cette version apporte une modification majeure du code qui utilise openCypher avec l'authentification IAM. Jusqu'à présent, la chaîne d'hôte contenue dans la signature IAM incluait le protocole `bolt://`, par exemple :

```
"Host":"bolt://(host URL):(port)"
```

À compter de la version du moteur `1.1.1.0`, ce protocole doit être omis :

```
"Host":"(host URL):(port)"
```

Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Améliorations de cette version du moteur

- Ajout de la prise en charge des types d'instances `x2g` basés sur Graviton2, types d'instances optimisés pour les charges de travail gourmandes en mémoire. Ils ne sont initialement disponibles qu'en quatre Régions AWS :
  - USA Est (Virginie du Nord) (`us-east-1`)
  - USA Est (Ohio) (`us-east-2`)
  - USA Ouest (Oregon) (`us-west-2`)
  - Europe (Irlande) (`eu-west-1`)

  Pour plus d'informations, consultez la page Tarification Neptune.
- Performances améliorées des étapes de Gremlin impliquant plusieurs traversées d'arêtes ou de sommets, recherches de propriétés ou recherches d'étiquettes.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin lié au traitement de l'étape `otherV()` dans une traversée enfant.
- Correction d'un bogue Gremlin dans les requêtes où `union` n'avait que des étapes de filtrage comme enfants. Par exemple :

```
g.V().union(has("name"), out("knows")).out()
```

- Correction d'un bogue SPARQL où les variables utilisées dans des expressions `FILTER` imbriquées dans des clauses `UNION` se voyaient attribuer des informations de portée non valides.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.1.1.0.R3, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.4`

- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.1.1.0.R3

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.1.1.0`.

Mise à niveau vers cette version

Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera indisponible à ce stade pendant environ six minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`

- DB instance shutdown
- Finished applying off-line patches to DB instance
- DB instance restarted

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version de maintenance d'Amazon Neptune, version 1.1.1.0.R2 (16/05/2022)

Depuis le 16 mai 2022, la version 1.1.1.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

> **Important**
>
> La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
> Afin de mener à bien la mise à niveau, chaque sous-réseau de chaque zone de disponibilité (AZ) doit disposer d'au moins une adresse IP disponible par instance Neptune. Par exemple, s'il existe une instance d'enregistreur et deux instances de lecteur dans le sous-réseau 1, et deux instances de lecteur dans le sous-réseau 2, le sous-réseau 1 doit avoir au moins trois adresses IP libres, tandis que le sous-réseau 2 doit avoir au moins deux adresses IP libres avant de commencer la mise à niveau.
> Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
> Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera alors indisponible pendant quelques minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
> Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> Cette version apporte une modification majeure du code qui utilise openCypher avec l'authentification IAM. Jusqu'à présent, la chaîne d'hôte contenue dans la signature IAM incluait le protocole `bolt://`, par exemple :
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> À compter de la version du moteur `1.1.1.0`, ce protocole doit être omis :
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> Pour obtenir des exemples, consultez Utilisation du protocole Bolt.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.1.1.0.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Première version de Gremlin prise en charge : `3.5.2`
- Dernière version de Gremlin est prise en charge : `3.5.4`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.1.1.0.R2

Votre cluster sera automatiquement mis à niveau vers cette version de correctif de maintenance lors de la prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.1.1.0`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera indisponible à ce stade pendant environ six minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`

- DB instance shutdown
- Finished applying off-line patches to DB instance
- DB instance restarted

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.1.0.0 du moteur Amazon Neptune (19/11/2021)

Depuis le 19 novembre 2021, la version 1.1.1.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
Afin de mener à bien la mise à niveau, chaque sous-réseau de chaque zone de disponibilité (AZ) doit disposer d'au moins une adresse IP disponible par instance Neptune. Par exemple, s'il existe une instance d'enregistreur et deux instances de lecteur dans le sous-réseau 1, et deux instances de lecteur dans le sous-réseau 2, le sous-réseau 1 doit avoir au moins trois adresses IP libres, tandis que le sous-réseau 2 doit avoir au moins deux adresses IP libres avant de commencer la mise à niveau.
Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera alors indisponible pendant quelques minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> À partir de cette version du moteur, Neptune ne prend plus en charge les types d'instances R4. Si vous utilisez une instance R4 dans votre cluster de bases de données, vous devrez la remplacer manuellement par un autre type d'instance avant de passer à cette version. Si vous utilisez une instance d'enregistreur R4, suivez ces instructions pour la déplacer.

Versions de correctifs ultérieures pour cette version

- Version de maintenance 1.1.0.0.R2 (16/05/2022)
- Version de maintenance 1.1.0.0.R3 (23/12/2022)

Nouvelles fonctionnalités pour cette version du moteur

- Introduction d'instances de base de données T4g polyvalentes et R6g optimisées pour la mémoire, alimentées par le processeur AWS Graviton2. Les instances basées sur Graviton2 offrent des performances nettement price/performance supérieures aux instances x86 comparables de génération actuelle pour diverses charges de travail. Les applications fonctionnent normalement sur ces nouveaux types d'instances. Il n'est donc pas nécessaire de transférer le code des applications lors de la mise à niveau vers ces nouveaux types d'instances.

Pour plus d'informations sur la disponibilité et la tarification d'une région, consultez la page Tarification Amazon Neptune.

- Ajout des modèles personnalisés dans Neptune ML.
- Ajout de la prise en charge des requêtes d'inférence SPARQL dans Neptune ML.
- Ajout d'un nouveau point de terminaison de flux pour les données du graphe de propriétés, à savoir :

```
https://Neptune-DNS:8182/propertygraph/stream
```

Le format de sortie de ce point de terminaison, nommé `PG_JSON`, est exactement le même que le format `GREMLIN_JSON` généré par l'ancien flux `gremlin/stream`.

Ce nouveau point de terminaison `propertygraph/stream` étend la prise en charge des flux Neptune à openCypher et remplace le point de terminaison `gremlin/stream` par le format de sortie `GREMLIN_JSON` associé.

Améliorations de cette version du moteur

- Améliorations apportées aux flux Neptune :
  - Ajout d'un champ `commitTimestamp` à l'objet `records` dans le format de réponse du journal des modifications de flux Neptune afin de fournir un horodatage pour chaque enregistrement d'un flux de journal des modifications.
  - Ajout d'une valeur `LATEST` au paramètre `iteratorType`, vous permettant de récupérer le dernier ID d'événement valide dans les flux. Consultez Appel de l'API Streams.
- Ajout de la prise en charge du score de confiance de l'inférence dans les requêtes de classification et de régression des nœuds Gremlin.
- Ajout de la prise en charge de la clause `OPTIONAL MATCH` dans openCypher.
- Ajout de la prise en charge de la clause `MERGE` dans openCypher.
- Ajout de la prise en charge de l'utilisation de `ORDER BY` dans les clauses `WITH` dans openCypher.
- Ajout de la prise en charge de la compréhension des modèles dans openCypher et ajout d'une prise en charge étendue de l'expression des modèles au-delà de la vérification de leur existence.
- Prise en charge étendue des clauses `DELETE` et `DELETE DETACH` dans openCypher, de sorte qu'elles peuvent désormais être utilisées avec d'autres clauses de mise à jour.

- Prise en charge étendue des clauses `CREATE` et `UPDATE` utilisées avec `RETURN` dans openCypher.
- Ajout dans le moteur DFE de la prise en charge des étapes Gremlin `limit`, `range` et `skip`.
- Amélioration de l'exécution des requêtes dans le moteur DFE lorsque ni `explain`, ni `profile` ne sont demandés.
- Amélioration de l'exécution des requêtes dans le moteur DFE pour l'expression `value`.
- Amélioration de divers modèles d'insertion conditionnelle Gremlin enchaînés afin d'éviter les exceptions de modification simultanée et de permettre le chaînage de modèles de requête tels que ceux-ci :
  - Insertion conditionnelle de sommets par ID, par exemple :

    ```
    g.V(ID).fold().coalesce(unfold(), g.addV("L1").property(id,ID))
    ```

  - Insertion conditionnelle de sommets avec plusieurs étiquettes, par exemple :

    ```
    g.V(ID).fold().coalesce(unfold(), g.addV("L1::L2").property(id,ID))
    ```

  - Insertion conditionnelle d'arêtes par ID, par exemple :

    ```
    g.E(ID).fold().coalesce(unfold(), V(from).addE(label).to(V(to)).property(id, ID))
    ```

  - Insertion conditionnelle d'arêtes avec plusieurs étiquettes, par exemple :

    ```
    g.E(ID).fold().coalesce(unfold(),
     g.addE(label).from(V(from)).to(V(to)).property(id, ID))
    ```

  - Insertion conditionnelle suivie d'une requête, par exemple :

    ```
    g.V(ID).fold().coalesce(unfold(),
     g.addV("L1").property(id,ID)).project("myvalues").by(valueMap())
    ```

  - Insertion conditionnelle avec propriétés ajoutées, par exemple :

    ```
    g.V(ID).fold().coalesce(unfold(),
     g.addV("L1").property(id,ID).property("name","pumba"))
    ```

Défauts corrigés dans cette version du moteur

- Désactivation de la fonctionnalité de statistiques sur les types d'instances `T3.medium`, qui n'étaient pas en mesure de la prendre en charge.
- Correction d'un bogue SPARQL lié à `explain` où une fonction `IN` utilisait des valeurs non constantes.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.1.0.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.11`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.1.0.0

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

La mise à niveau vers cette version n'est pas automatique.

Mise à niveau vers cette version

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Au lieu d'`--apply-immediately`, vous pouvez spécifier `--no-apply-immediately`. Pour effectuer une mise à niveau de version majeure, le allow-major-version-upgrade paramètre est obligatoire. Assurez-vous également d'inclure la version du moteu. Dans le cas contraire, le moteur sera peut-être mis à niveau vers une autre version.

Si votre cluster utilise un groupe de paramètres de cluster personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

De même, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé, veillez à inclure ce paramètre pour le spécifier :

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version de maintenance d'Amazon Neptune, version 1.1.0.0.R3 (23/12/2022)

Depuis le 23 décembre 2023, la version 1.1.0.0.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
Afin de mener à bien la mise à niveau, chaque sous-réseau de chaque zone de disponibilité (AZ) doit disposer d'au moins une adresse IP disponible par instance Neptune. Par exemple, s'il existe une instance d'enregistreur et deux instances de lecteur dans le sous-réseau 1, et deux instances de lecteur dans le sous-réseau 2, le sous-réseau 1 doit avoir au moins trois adresses IP libres, tandis que le sous-réseau 2 doit avoir au moins deux adresses IP libres avant de commencer la mise à niveau.
Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera alors indisponible pendant quelques minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`

- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Améliorations de cette version du moteur

- Amélioration des performances et corrections de divers opérateurs Gremlin, notamment `repeat`, `coalesce`, `store` et `aggregate`.

Défauts corrigés dans cette version du moteur

- Correction d'un problème de pic d'activité du CPU.
- Correction d'un bogue openCypher où les requêtes renvoyaient la chaîne `"null"` au lieu d'une valeur nulle dans Bolt et SPARQL-JSON.
- Correction d'un bogue du journal d'audit qui entraînait la journalisation d'informations inutiles et l'absence de certains champs dans les journaux.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.1.0.0.R3, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.11`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.1.0.0.R3

Votre cluster sera automatiquement mis à niveau vers cette version de correctif de maintenance lors de la prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.1.0.0`.

Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera indisponible à ce stade pendant environ six minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

À partir de cette version du moteur, Neptune ne prend plus en charge les types d'instances R4. Si vous utilisez une instance R4 dans votre cluster de bases de données, vous devrez

la remplacer manuellement par un autre type d'instance avant de passer à cette version. Si vous utilisez une instance d'enregistreur R4, suivez ces instructions pour la déplacer.

Mise à niveau vers cette version

Amazon Neptune 1.1.0.0.R3 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version de maintenance d'Amazon Neptune, version 1.1.0.0.R2 (16/05/2022)

Depuis le 16 mai 2022, la version 1.1.0.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Important

La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.
Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera alors indisponible pendant quelques minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :

- `Applying off-line patches to DB instance`
- `DB instance shutdown`
- `Finished applying off-line patches to DB instance`
- `DB instance restarted`

Défauts corrigés dans cette version du moteur

- Correction d'un bogue en raison duquel le cache d'informations d'identification interne n'était pas correctement vidé pour les points de terminaison autres que ceux des requêtes, tels que le point de terminaison d'état.
- Correction d'un bogue qui augmentait le retard de réplication après une mise à niveau du moteur.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.1.0.0.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.11`
- Version d'openCypher : `Neptune-9.0.20190305-1.0`
- Version de SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.1.0.0.R2

Votre cluster sera automatiquement mis à niveau vers cette version de correctif de maintenance lors de la prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.1.0.0`.

> **Important**
>
> La mise à niveau vers cette version du moteur à partir d'une version antérieure à **`1.1.0.0`** déclenche également une mise à niveau du système d'exploitation sur toutes les instances de votre cluster de bases de données. Étant donné que les demandes d'écriture actives qui se produisent pendant la mise à niveau du système d'exploitation ne sont pas traitées, vous devez suspendre toutes les charges de travail d'écriture dans le cluster en cours de mise à niveau, y compris les chargements de données en bloc, avant de démarrer la mise à niveau.

Au début de la mise à niveau, Neptune génère un instantané dont le nom est composé de `preupgrade` suivi d'un identifiant généré automatiquement en fonction des informations de votre cluster de bases de données. Cet instantané ne vous est pas facturé. Vous pouvez l'utiliser pour restaurer votre cluster de bases de données en cas de problème pendant le processus de mise à niveau.
Lorsque la mise à niveau du moteur elle-même sera terminée, la nouvelle version du moteur sera brièvement disponible sur l'ancien système d'exploitation, mais en moins de cinq minutes, toutes les instances de votre cluster commenceront simultanément une mise à niveau du système d'exploitation. Votre cluster de bases de données sera indisponible à ce stade pendant environ six minutes. Vous pourrez reprendre les charges de travail d'écriture une fois la mise à niveau terminée.
Ce processus génère les événements suivants :

- Messages d'événements par cluster :
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Messages d'événements par instance :
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> À partir de cette version du moteur, Neptune ne prend plus en charge les types d'instances R4. Si vous utilisez une instance R4 dans votre cluster de bases de données, vous devrez la remplacer manuellement par un autre type d'instance avant de passer à cette version. Si vous utilisez une instance d'enregistreur R4, suivez ces instructions pour la déplacer.

Mise à niveau vers cette version

Amazon Neptune 1.1.0.0.R2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.5.1 du moteur Amazon Neptune (01/10/2021)

Depuis le 1er octobre 2021, la version 1.0.5.1 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Versions de correctifs ultérieures pour cette version

- Sortie : 1.0.5.1.R2 (26/10/2021)
- Sortie : 1.0.5.1.R3 (13/01/2022)
- Version de maintenance 1.0.5.1.R4 (16/05/2022)

Nouvelles fonctionnalités pour cette version du moteur

- Ajout d'un cache de résultats pour mettre en cache les résultats des requêtes spécifiées.
- Date/time Support ajouté dans Neptune OpenCypher.
- Ajout de la prise en charge de l'accès `List` et `Map` aux éléments dans Neptune openCypher.

Améliorations de cette version du moteur

- Les noms des points de terminaison Neptune openCypher ne sont pas sensibles à la casse.
- Amélioration de la fonctionnalité explain d'openCypher.
- Amélioration des modèles de requêtes Gremlin à upsert unique se terminant par les étapes `iterate()` et `profile()`.
- Amélioration des performances des fonctions Gremlin `keys()` et `property()`.
- L'étape Gremlin `dedup()` est exécutée dans le DFE lorsqu'elle est utilisée avec une portée globale.
- Les prédicats HAS Gremlin suivants sont exécutés dans le moteur DFE lorsque celui-ci est activé :
  - `EQ`
  - `NEQ`
  - `LT`
  - `LTE`
  - `GT`
  - `GTE`

  - `BETWEEN`
  - `INSIDE`
  - `OUTSIDE`
  - `WITHIN`
  - `AND (connectives)`
  - `OR (connectives)`
- Amélioration des performances de requêtes LIMIT.
- Amélioration des performances des requêtes d'agrégation générales openCypher.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin qui permettait à une arête d'être connectée à une autre arête.
- Correction d'un bogue Gremlin qui entraînait le choix d'une stratégie de jointure sous-optimale.
- Correction d'un bogue Gremlin qui bloquait la sérialisation des nœuds et des relations lorsque plus de 100 propriétés étaient présentes.
- Correction d'un bogue qui ralentissait la planification de l'exécution des requêtes comportant de grands modèles de graphes.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.5.1, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.11`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.5.1

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

La mise à niveau vers cette version n'est pas automatique.

Mise à niveau vers cette version

Amazon Neptune 1.0.5.1 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations,

l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version de maintenance d'Amazon Neptune, version 1.0.5.1.R4 (16/05/2022)

Depuis le 16 mai 2022, la version 1.0.5.0.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.5.1.R4, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.11`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.5.1.R4

Votre cluster sera automatiquement mis à niveau vers cette version de correctif de maintenance lors de la prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.5.1`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.5.1.R4 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
```

```
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.5.1.R3 du moteur Amazon Neptune (13/01/2022)

Depuis le 13 janvier 2022, la version 1.0.5.1.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue qui pouvait provoquer une fuite de ressources lorsqu'une requête ne parvenait pas à obtenir toutes les ressources dont elle avait besoin.
- Correction d'une petite fuite de mémoire lors de l'exécution d'une requête causée par une allocation de mémoire non revendiquée.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.5.1.R3, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.11`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.5.1.R3

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.5.1`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.5.1.R3 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.5.1.R2 du moteur Amazon Neptune (26/10/2021)

Depuis le 26 octobre 2021, la version 1.0.5.1.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue qui provoquait le redémarrage du serveur lorsqu'une erreur transitoire se produisait pendant de la création d'une ancienne version d'un élément de graphe, dans le cadre d'un isolement de lecture reproductible. Neptune renvoie désormais une erreur à la place, afin que le client puisse effectuer une nouvelle tentative.
- Correction d'un bogue qui provoquait le redémarrage du serveur lorsqu'une erreur transitoire se produisait lors de la mise à jour d'une seule cardinalité. Neptune renvoie désormais une erreur à la place, afin que le client puisse effectuer une nouvelle tentative.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.5.1.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.11`

- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.5.1.R2

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.5.1`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.5.1.R2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.5.0 du moteur Amazon Neptune (27/07/2021)

Depuis le 27 juillet 2021, la version 1.0.5.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Versions de correctifs ultérieures pour cette version

- Sortie : 1.0.5.0.R2 (16/08/2021)
- Sortie : 1.0.5.0.R3 (15/09/2021)
- Version de maintenance 1.0.5.0.R5 (16/05/2022)

Nouvelles fonctionnalités pour cette version du moteur

- Neptune ML a été publié pour une utilisation en production avec de nombreuses nouvelles fonctionnalités et n'est plus en mode laboratoire.
- Ajout de la prise en charge initiale du langage de requête openCypher, en mode expérimental. openCypher est le standard open source pour le langage de requête Cypher. Sa syntaxe est spécifiée dans le Cypher Query Language Reference (version 9) et est gérée par le projet openCypher.

  Consultez Accès au graphe Neptune avec openCypher pour plus d'informations sur l'implémentation Neptune de ce langage.

Le protocole Bolt, que les clients Neptune utilisent pour les requêtes openCypher, est également pris en charge. Consultez Utilisation du protocole Bolt pour envoyer des requêtes openCypher à Neptune.

La prise en charge d'openCypher est désormais activée automatiquement, mais dépend du Moteur DFE Neptune, qui n'est actuellement disponible qu'en mode laboratoire. Le paramètre `DFEQueryEngine` par défaut du cluster de bases de données `neptune_lab_mode` est désormais `DFEQueryEngine=viaQueryHint` : le moteur est donc activé, mais uniquement utilisé pour les requêtes dont l'indicateur de requête `useDFE` est défini sur `true`. Si vous désactivez le moteur DFE en définissant `DFEQueryEngine=disabled`, vous ne pouvez pas utiliser openCypher.

- Ajout de la prise en charge du protocole HTTP SPARQL 1.1 Graph Store. Consultez Utilisation du protocole HTTP SPARQL 1.1 Graph Store (GSP) dans Amazon Neptune.
- Remplacement du paramètre de mode expérimental par défaut du Moteur DFE Neptune par `viaQueryHint` : le moteur DFE est désormais activé par défaut, mais uniquement utilisé pour les requêtes dont l'indicateur de requête `useDFE` est défini sur `true`.
- Ajout d'une nouvelle CloudWatch métrique Amazon,`StatsNumStatementsScanned`, pour surveiller le calcul des statistiques pour le moteur Neptune DFE. Consultez Utilisation de la `StatsNumStatementsScanned` CloudWatch métrique pour surveiller le calcul des statistiques.

Améliorations de cette version du moteur

- Ajout du support pour Apache TinkerPop 3.4.11.

> **Important**
>
> Une modification a été apportée à TinkerPop la version 3.4.11 qui améliore l'exactitude du traitement des requêtes, mais qui, pour le moment, peut parfois avoir un impact sérieux sur les performances des requêtes.
> Par exemple, une requête de ce type peut être beaucoup plus lente :
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
>   out()
> ```

Les sommets après le pas de limite sont désormais récupérés de manière non optimale en raison de la modification 3.4.11. TinkerPop Pour éviter cela, vous pouvez modifier la requête en ajoutant l'étape barrier() à tout moment après `order().by()`. Par exemple :

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

- L'indicateur de requête `joinOrder` SPARQL est désormais pris en charge par l'autre moteur de requête Neptune DFE.
- La sortie de l'API d'état Neptune a été étendue et réorganisée afin de clarifier les paramètres et les fonctionnalités du cluster de bases de données.

  La nouvelle sortie contient un objet `features` de niveau supérieur comportant des informations d'état sur les fonctionnalités du cluster de bases de données et un objet `settings` de niveau supérieur comportant des informations sur les paramètres. Pour examiner le nouveau format, consultez Exemple de sortie de la commande instance status.
- La gestion des journaux de modifications des flux a été améliorée lorsque des flux `AFTER_SEQUENCE_NUMBER` sont demandés avec le dernier ID d'événement sur le serveur, quand cet ID a déjà expiré. Le serveur ne renvoie plus d'erreur d'identifiant d'événement ayant expiré si l'identifiant d'événement demandé est le dernier identifiant d'événement purgé sur le serveur.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin lié à l'ordre des valeurs numériques.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.5.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.11`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.5.0

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

La mise à niveau vers cette version n'est pas automatique.

Mise à niveau vers cette version

Amazon Neptune 1.0.5.0 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise

à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the instance before
>   proceeding with the upgrade.
> ```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version de maintenance d'Amazon Neptune, version 1.0.5.0.R5 (16/05/2022)

Depuis le 16 mai 2022, la version 1.0.5.0.R5 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.5.0.R5, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.11`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.5.0.R5

Votre cluster sera automatiquement mis à niveau vers cette version de correctif maintenance lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur 1.0.5.0.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.5.0.R5 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez

mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.5.0.R3 du moteur Amazon Neptune (15/09/2021)

Depuis le 15 septembre 2021, la version 1.0.5.0.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue qui empêchait le moteur de répondre dans l'une des situations suivantes :
  - Un chargement en bloc se produit en même temps que le calcul automatique des statistiques.
  - Un calcul de statistiques a été demandé manuellement alors qu'un tel calcul était déjà en cours.
- Correction d'un bogue lié à la détection des blocages et à l'acquisition des blocages qui pouvait provoquer un incident au niveau du moteur.
- Correction d'un bogue Gremlin où le moteur renvoyait une erreur lorsqu'il trouvait des données inconnues provenant d'un point de terminaison ML distant dans une requête d'inférence Gremlin.
- Correction de plusieurs bogues dans la gestion des modèles ML APIs liés aux tâches de transformation des modèles et aux recommandations d'instances.
- Correction d'un bug qui pouvait provoquer le crash du moteur lors de la génération du nœud et du bord IDs.
- Correction d'un bogue qui ralentissait la génération de plans de requêtes pour les requêtes comportant de grands modèles de graphes.
- Correction d'un bogue openCypher qui pouvait provoquer le blocage d'une requête lors de la récupération d'un nœud comportant plus de 100 propriétés.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.5.0.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.11`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.5.0.R3

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur 1.0.5.0.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.5.0.R3 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.5.0.R2 du moteur Amazon Neptune (16/08/2021)

Depuis le 16 août 2021, la version 1.0.5.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

- Désactivation d'une optimisation effectuée dans la version de moteur `1.0.5.0` qui permettait au cache de recherche Neptune de survivre aux redémarrages du moteur sur les réplicas. Les redémarrages de réplicas vident à présent le cache de recherche.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.5.0.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.11`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.5.0.R2

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.5.0`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.5.0.R2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez

mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.4.2 du moteur Amazon Neptune (01/06/2021)

> **Note**
>
> La version 1.0.4.2.R2 du moteur a été la première version de 1.0.4.2 à être publiée.

Rubriques

- Version 1.0.4.2.R5 du moteur Amazon Neptune (16/08/2021)
- Version 1.0.4.2.R4 du moteur Amazon Neptune (23/07/2021)
- Version 1.0.4.2.R3 du moteur Amazon Neptune (28/06/2021)
- Version 1.0.4.2.R2 du moteur Amazon Neptune (01/06/2021)
- Version 1.0.4.2.R1 du moteur Amazon Neptune (27/05/2021)

Version 1.0.4.2.R5 du moteur Amazon Neptune (16/08/2021)

Depuis le 16 août 2021, la version 1.0.4.2.R5 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

- Désactivation d'une optimisation effectuée dans la version de moteur `1.0.4.2.R4` qui permettait au cache de recherche Neptune de survivre aux redémarrages du moteur sur les réplicas. Les redémarrages de réplicas vident à présent le cache de recherche.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.4.2.R5, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.10`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.4.2.R5

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.4.2`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Version 1.0.4.2.R4 du moteur Amazon Neptune (23/07/2021)

Depuis le 23 juillet 2021, la version 1.0.4.2.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Améliorations de cette version du moteur

- Amélioration du comportement du cache de recherche afin d'éviter l'effacement redondant du cache après une réinitialisation rapide sur un réplica.
- Gestion améliorée des journaux de modifications des flux lorsque les flux `AFTER_SEQUENCE_NUMBER` sont demandés avec le dernier ID d'événement sur le serveur, quand cet ID a déjà expiré. Le serveur ne renvoie plus d'erreur d'identifiant d'événement ayant expiré si l'identifiant d'événement demandé est le dernier identifiant d'événement purgé sur le serveur.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue introduit dans la version 1.0.4.0.R1 où les requêtes ne renvoyaient pas l'intégralité des valeurs de chaîne supérieures à 760 caractères. Les termes concernés par ce bogue étaient les littéraux RDF et/ou URIs, ou Gremlin IDs, les clés et les valeurs de chaîne.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.4.2.R4, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.10`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.4.2.R4

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.4.2`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Version 1.0.4.2.R3 du moteur Amazon Neptune (28/06/2021)

Depuis le 28 juin 2021, la version 1.0.4.2.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Problèmes connus dans cette version du moteur

Problème :

Bogue SPARQL qui ne respecte pas le type de média dans un en-tête `Accept`en cas d'espaces.

Par exemple, une requête `-H "Accept: text/csv; q=1.0, */*; q=0.1"` renvoie une sortie JSON plutôt qu'une sortie CSV.

Solution :

Si vous supprimez les espaces dans la clause `Accept` de l'en-tête, le moteur renvoie le résultat dans le format demandé correct. En d'autres termes, au lieu de `-H "Accept: text/csv; q=1.0, */*; q=0.1"` , utilisez :

```
-H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

Défauts corrigés dans cette version du moteur

- Correction d'un bogue lors de l'effacement du cache de recherche sur les réplicas après une réinitialisation rapide.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.4.2.R3, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.10`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.4.2.R3

Cette version de correctif est facultative, sauf si votre cluster de bases de données utilise une ou plusieurs instances `R5d`. Si votre cluster possède des instances `R5d`, il sera automatiquement mis à niveau lors de la prochaine fenêtre de maintenance. Dans le cas contraire, sa mise à niveau vers cette version de correctif ne sera pas automatique.

Vous pouvez mettre à jour `1.0.4.2.R2` la version vers cette `1.0.4.2.R3` version manuellement à l'aide de la AWS CLI apply-pending-maintenance-actioncommande (l'ApplyPendingMaintenanceActionAPI).

Version 1.0.4.2.R2 du moteur Amazon Neptune (01/06/2021)

Depuis le 1er juin 2021, la version 1.0.4.2.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Versions de correctifs ultérieures pour cette version

- Sortie : 1.0.4.2.R3 (28/06/2021)

Problèmes connus dans cette version du moteur

Problème :

Bogue SPARQL qui ne respecte pas le type de média dans un en-tête `Accept`en cas d'espaces.

Par exemple, une requête `-H "Accept: text/csv; q=1.0, */*; q=0.1"` renvoie une sortie JSON plutôt qu'une sortie CSV.

Solution :

Si vous supprimez les espaces dans la clause `Accept` de l'en-tête, le moteur renvoie le résultat dans le format demandé correct. En d'autres termes, au lieu de `-H "Accept: text/csv; q=1.0, */*; q=0.1"` , utilisez :

```
-H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

Nouvelles fonctionnalités pour cette version du moteur

- Ajout du nouveau type d'instance R5d, qui inclut un cache de recherche pour accélérer les lectures dans les cas d'utilisation impliquant un volume élevé de recherche de valeurs de propriété ou de littéraux RDF. Consultez Le cache de recherche Neptune peut accélérer les requêtes de lecture.
- Ajout d'un nouveau paramètre en mode laboratoire qui permet au moteur DFE expérimental d'être invoqué uniquement par requête avec l'indicateur de requête `useDFE`.

Améliorations de cette version du moteur

- Ajout du support pour la version TinkerPop 3.4.10.
- Ajout de la prise en charge de l'utilisation de l'étape de configuration `withStrategies( )` lors de l'envoi de requêtes de script Gremlin. Plus précisément, les étapes `SubgraphStrategy`, `PartitionStrategy`, `ReadOnlyStrategy`, `EdgeLabelVerificationStrategy` et `ReservedKeysVerificationStrategy` sont toutes prises en charge.
- Ajout de l'optimisation des traversées `V()` au milieu d'une requête. Auparavant, ces traversées n'étaient pas optimisées dans Neptune.
- Ajout de la prise en charge de la RFC 2141 URNs à utiliser comme `namedGraphUri` paramètres `baseUri` et pour un chargement en vrac.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin dans l'analyseur dans lequel les requêtes incorrectes étaient considérées comme valides.
- Correction d'un bogue Gremlin où le déploiement d'un effet secondaire `aggregate()` avec `cap().unfold()` pour le remplacer par `valueMap()` provoquait une exception.
- Correction d'un bogue Gremlin qui provoquait l'échec de certaines étapes `property()` après une étape `addV()` avec l'erreur « cannot cast to String » (Conversion en chaîne impossible).
- Correction d'un bogue Gremlin qui empêchait certains modèles d'insertion conditionnelle de déclencher des exceptions de modification simultanée.
- Correction d'un bogue Gremlin de sorte que le délai d'expiration de la demande de requête ne puisse plus dépasser le délai d'expiration de la session.

- Correction d'un bogue SPARQL en raison duquel les mises à jour utilisant LOAD ou UNLOAD pouvaient échouer avec un code HTTP 500 au lieu du code HTTP 400 lorsque le serveur distant n'était pas disponible.
- Correction d'un bogue où les appels d'API de flux échouaient lorsque des valeurs `commitNum` ou `opNum` supérieures à la limite d'entiers signés de 32 bits (2 147 483 647) étaient utilisées.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.4.2.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.10`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.4.2.R2

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

La mise à niveau vers cette version n'est pas automatique.

Mise à niveau vers cette version

Amazon Neptune 1.0.4.2.R2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.2 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.2 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un

instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.4.2.R1 du moteur Amazon Neptune (27/05/2021)

La version 1.0.4.1.R1 du moteur n'a jamais été déployée.

Version 1.0.4.1 du moteur Amazon Neptune (08/12/2020)

Depuis le 8 décembre 2020, la version 1.0.4.1 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Versions de correctifs ultérieures pour cette version

- Sortie : 1.0.4.1.R1.1 (22/03/2021)
- Sortie : 1.0.4.1.R2 (24/02/2021)

> **Important**
>
> Sortie : 1.0.4.0 (12/10/2020) a rendu les protocoles TLS 1.2 et HTTPS obligatoires pour toutes les connexions à Amazon Neptune. Cependant, un bogue présent dans cette version a permis aux connexions HTTP (connexions TLS and/or obsolètes) de continuer à fonctionner pour les clients qui avaient précédemment défini un paramètre de cluster de base de données afin d'empêcher l'application des connexions HTTPS.
> Ce bogue a été corrigé dans les versions de correctifs 1.0.4.0.R2 et 1.0.4.1.R2, mais le correctif provoquait des échecs de connexion inattendus lors de l'installation automatique des correctifs. Pour cette raison, les deux correctifs ont été annulés et ne peuvent être installés que manuellement, afin de vous permettre de mettre à jour votre configuration pour TLS 1.2.
> Le fait de devoir l'utiliser SSL/TLS pour toutes les connexions à Neptune affecte vos connexions avec la console Gremlin, le pilote Gremlin, Python, .NET, NodeJS, REST, ainsi que les connexions à l'équilibreur de charge. APIs Si vous avez utilisé le protocole HTTP ou une ancienne version de TLS jusqu'à présent, vous devrez mettre à jour le client et les pilotes concernés, et modifier le code pour qu'il utilise exclusivement le protocole HTTPS avant d'installer les derniers correctifs sur le système.

Nouvelles fonctionnalités pour cette version du moteur

- Présentation de la fonctionnalité Neptune ML, qui apporte de puissantes fonctionnalités de machine learning à Amazon Neptune. Consultez Amazon Neptune ML pour le machine learning sur les graphes.
- Ajout d'une opération `UNLOAD` SPARQL personnalisée pour supprimer les données extraites d'une source distante. Consultez SPARQL UPDATE UNLOAD.

Améliorations de cette version du moteur

- Optimisation de certains modèles d'insertion conditionnelle Gremlin pour éviter les exceptions de modification simultanée.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin qui pouvait entraîner des résultats manquants pour un modèle spécifique de requêtes qui utilisaient l'étape `as()`.
- Correction d'un bogue Gremlin qui pouvait provoquer des erreurs lors de l'utilisation de l'étape `project()` imbriquée dans une autre étape, comme `union()`.
- Correction d'un bogue Gremlin dans l'étape `project()`.
- Correction d'un bogue Gremlin lié à la traversée basée sur des chaînes, à cause duquel l'étape `none()` ne fonctionnait pas.
- Correction d'un bogue Gremlin lié à la traversée basée sur des chaînes, à cause duquel un mappage vide n'était pas pris en charge comme argument de l'étape `inject()`.
- Correction d'un bogue Gremlin lié à l'exécution de traversées basées sur des chaînes dans le moteur DFE et à cause duquel une méthode de terminal telle que `toList()` ne fonctionnait pas correctement.
- Correction d'un bogue Gremlin qui empêchait de clôturer des transactions lors de l'utilisation de l'étape `iterate()` dans les requêtes sous forme de chaîne.
- Correction d'un bogue Gremlin qui pouvait provoquer une exception pour les requêtes utilisant le modèle `is(P.gte(0))` dans certaines situations.
- Correction d'un bogue Gremlin qui pouvait provoquer une exception pour les requêtes utilisant le modèle `order().by(T.id)` dans certaines situations.
- Correction d'un bogue Gremlin en raison duquel les requêtes utilisant le modèle `addV().aggregate()` généraient des résultats incorrects dans certaines situations.
- Correction d'un bogue Gremlin qui pouvait provoquer une exception dans certaines situations pour les requêtes utilisant l'étape `path()` suivie du modèle d'étape `project()`.
- Correction d'un bogue SPARQL où la fonction `SUBSTR` signalait une erreur au lieu de renvoyer une chaîne vide.
- Correction d'un bogue dans le moteur DFE en raison duquel les opérations de jointure dans les plans de requêtes non bloquants pouvaient générer des résultats incorrects en présence de variables indépendantes.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.4.1, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.8`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.4.1

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.4.1`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.4.1 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.4.1.R1.1 du moteur Amazon Neptune (22/03/2021)

Depuis le 22 mars 2021, la version 1.0.4.1.R1.1 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

- Désactivation d'une optimisation pour les modèles d'insertion conditionnelle Gremlin qui peuvent être ajoutés à des étiquettes et à des propriétés existantes.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.4.1.R1.1, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.8`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.4.1.R1.1

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.4.1`.

Mise à niveau vers cette version

Amazon Neptune 1.0.4.1.R1.1 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations,

l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.4.1.R2 du moteur Amazon Neptune (24/02/2021)

Depuis le 24 février 2021, la version 1.0.4.1.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Versions de correctifs ultérieures pour cette version

- Sortie : 1.0.4.1.R2.1 (11/03/2021)

Nouvelles fonctionnalités pour cette version du moteur

- Neptune prend en charge la compression des fichiers individuels au format `bzip2` pour les chargements en bloc. Consultez Formats de chargement de données.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue dans Sortie : 1.0.4.0 (12/10/2020) qui autorisait les connexions à Neptune à l'aide de TLS `HTTP` ou version antérieure, plutôt que `HTTPS` et TLS 1.2.

  > **Important**
  >
  > Le fait de devoir l'utiliser SSL/TLS pour toutes les connexions à Neptune peut être un changement radical. Cela affecte vos connexions avec la console Gremlin, le pilote Gremlin, Python Gremlin, .NET, NodeJS, APIs REST, ainsi que les connexions à l'équilibreur de charge. Si vous avez utilisé le protocole HTTP ou une ancienne version de TLS jusqu'à présent, vous devrez mettre à jour le client et les pilotes concernés avant d'installer ce correctif, et modifier le code pour utiliser exclusivement le protocole HTTPS.

- Correction d'un bogue Gremlin où l'exception `InternalFailureException` était définie comme code de réponse dans certaines circonstances lors d'un événement `ConcurrentModificationException`.
- Correction d'un bogue Gremlin où, dans certaines conditions, la mise à jour des arêtes ou des sommets pouvait provoquer une exception `InternalFailureException` transitoire.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.4.1.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.8`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.4.1.R2

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.4.1`.

Mise à niveau vers cette version

Amazon Neptune 1.0.4.1.R2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces

instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.4.1.R2.1 du moteur Amazon Neptune (11/03/2021)

Depuis le 11 mars 2021, la version 1.0.4.1.R2.1 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

- Désactivation d'une optimisation pour les modèles d'insertion conditionnelle Gremlin qui peuvent être ajoutés à des étiquettes et à des propriétés existantes.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.4.1.R2.1, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.8`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.4.1.R2.1

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.4.1.R2`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.4.1.R2.1 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1.R2 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1.R2 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise

à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the instance before
>   proceeding with the upgrade.
> ```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.4.0 du moteur Amazon Neptune (12/10/2020)

Depuis le 12 octobre 2020, la version 1.0.4.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Versions de correctifs ultérieures pour cette version

- Sortie : 1.0.4.0.R2 (24/02/2021)

Nouvelles fonctionnalités pour cette version du moteur

- Ajout de la compression au niveau des frames pour Gremlin.

Améliorations de cette version du moteur

- Amazon Neptune exige désormais l'utilisation du protocole SSL (Secure Sockets Layer) avec le protocole TLSv1 .2 pour toutes les connexions à Neptune dans toutes les régions, en utilisant ces suites de chiffrement puissantes :
  - `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
  - `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
  - `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
  - `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
  - `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
  - `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

Cela est vrai à la fois pour REST et pour WebSocket les connexions à Neptune, et cela signifie que vous devez utiliser le protocole HTTPS plutôt que le protocole HTTP lorsque vous vous connectez à Neptune dans toutes les régions.

Étant donné que les connexions client utilisant HTTP ou TLS 1.1 ne seront plus prises en charge nulle part, assurez-vous que vos clients et votre code ont été mis à jour pour pouvoir utiliser TLS 1.2 et HTTPS avant de passer à cette version du moteur.

> **Important**
>
> Le fait de devoir l'utiliser SSL/TLS pour toutes les connexions à Neptune peut être un changement radical. Cela affecte vos connexions avec la console Gremlin, le pilote Gremlin, Gremlin Python, .NET, NodeJS, APIs REST, ainsi que les connexions à l'équilibreur de charge. Si vous avez utilisé le protocole HTTP pour une partie ou la totalité de ces connexions, vous devez à présent mettre à jour le client et les pilotes concernés et modifier le code pour qu'il utilise le protocole HTTPS. Dans le cas contraire, les connexions échoueront.
> Un bogue dans cette version a permis aux connexions HTTP (connexions TLS and/or obsolètes) de continuer à fonctionner pour les clients qui avaient précédemment défini un paramètre de cluster de base de données afin d'empêcher l'application des connexions HTTPS. Ce bogue a été corrigé dans les versions de correctifs 1.0.4.0.R2 et 1.0.4.1.R2, mais le correctif provoquait des échecs de connexion inattendus lors de l'installation automatique des correctifs.
> Pour cette raison, les deux correctifs ont été annulés et ne peuvent être installés que manuellement, afin de vous permettre de mettre à jour votre configuration pour TLS 1.2.

- Mise à niveau TinkerPop vers la version 3.4.8. Il s'agit d'une mise à niveau rétrocompatible. Consultez le journal des TinkerPop modifications pour connaître les nouveautés.
- Amélioration des performances pour l'étape Gremlin `properties()`.
- Ajout de détails sur `BindOp` et `MultiplexerOp` dans les rapports d'explication et de profil.
- Ajout d'une fonction de prélecture des données pour améliorer les performances en cas de défaillance du cache.
- Ajout d'un nouveau paramètre `allowEmptyStrings` dans le paramètre `parserConfiguration` du chargeur en bloc. Il permet de traiter les chaînes vides comme des valeurs de propriété valides dans les chargements CSV (voir Paramètres de demande du chargeur Neptune).

- Le chargeur autorise désormais un point-virgule avec caractère d'échappement dans les colonnes CSV à valeurs multiples.

Défauts corrigés dans cette version du moteur

- Correction d'une fuite de mémoire Grenlin potentielle liée à l'étape `both()`.
- Correction d'un bogue en raison duquel les métriques des demandes manquaient parce qu'un point de terminaison se terminant par / n'était pas géré correctement.
- Correction d'un bogue en raison duquel les réplicas prenaient du retard et redémarraient sous forte charge lorsque le moteur DFE était activé en mode laboratoire.
- Correction d'un bogue qui empêchait le message d'erreur correct d'être signalé lorsqu'un chargement groupé échouait en raison d'une out-of-memory condition.
- Correction d'un bogue SPARQL en raison duquel l'encodage des caractères était placé dans l'en-tête Content-Encoding dans les réponses aux requêtes SPARQL. Au lieu de cela, `charset` est désormais placé dans l'en-tête Content-Type, ce qui permet aux clients HTTP de reconnaître le jeu de caractères utilisé automatiquement.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.4.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.8`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.4.0

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

La mise à niveau vers cette version n'est pas automatique.

Mise à niveau vers cette version

Amazon Neptune 1.0.4.0 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.4.0.R2 du moteur Amazon Neptune (24/02/2021)

Depuis le 24 février 2021, la version 1.0.4.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue dans Sortie : 1.0.4.0 (12/10/2020) qui autorisait les connexions à Neptune à l'aide de TLS HTTP ou version antérieure, plutôt que HTTPS et TLS 1.2.

  > **Important**
  >
  > Le fait de devoir l'utiliser SSL/TLS pour toutes les connexions à Neptune peut être un changement radical. Cela affecte vos connexions avec la console Gremlin, le pilote Gremlin, Python Gremlin, .NET, NodeJS, APIs REST, ainsi que les connexions à l'équilibreur de charge. Si vous avez utilisé le protocole HTTP ou une ancienne version de TLS jusqu'à présent, vous devrez mettre à jour le client et les pilotes concernés avant d'installer ce correctif, et modifier le code pour utiliser exclusivement le protocole HTTPS.

- Correction d'un bogue dans le chargement en bloc des fichiers CSV impliquant des libellés se terminant par #.
- Correction d'un bogue Gremlin où l'exception `InternalFailureException` était définie comme code de réponse dans certaines circonstances lors d'un événement `ConcurrentModificationException`.
- Correction d'un bogue Gremlin où, dans certaines conditions, la mise à jour des arêtes ou des sommets pouvait provoquer une exception `InternalFailureException` transitoire.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.4.0.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.8`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.4.0.R2

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.4.0`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.4.0.R2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise

à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the instance before
>   proceeding with the upgrade.
> ```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.3.0 du moteur Amazon Neptune (03/08/2020)

Depuis le 3 août 2020, la version 1.0.3.0 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Versions de correctifs ultérieures pour cette version

- Sortie : 1.0.3.0.R2 (12/10/2020)
- Sortie : 1.0.3.0.R3 (19/02/2021)

Nouvelles fonctionnalités pour cette version du moteur

- Neptune a introduit un nouveau moteur de requête alternatif (DFE) qui contribue à accélérer considérablement l'exécution des requêtes. Consultez Autre moteur de requête (DFE) Amazon Neptune.
- Ce DFE s'appuie sur des statistiques prégénérées concernant les données du graphe Neptune, qui sont gérées via de nouveaux points de terminaison de statistiques. Consultez Statistiques DFE.
- Vous pouvez désormais exclure les tâches de chargement en file d'attente de la liste des charges IDs renvoyées par l' Get-Status API Loader en définissant le nouveau `includeQueuedLoads` paramètre sur FALSE. Consultez Paramètres de demande du Neptune Loader Get-Status.
- Neptune prend désormais en charge les en-têtes de fin pour les réponses aux requêtes SPARQL, qui peuvent contenir un code d'erreur et un message si une demande échoue après avoir commencé à renvoyer des fragments de réponse. Consultez En-têtes de suivi HTTP facultatifs pour les réponses SPARQL en plusieurs parties.

- Neptune vous permet désormais également d'activer l'encodage des réponses fragmentées pour les requêtes Gremlin. Comme dans le cas de SPARQL, les segments de réponse ont des en-têtes de fin qui peuvent contenir un code d'erreur et un message en cas d'échec après que la requête a commencé à renvoyer des fragments de réponse. Consultez Utilisation d'en-têtes de suivi HTTP facultatifs pour activer les réponses Gremlin en plusieurs parties.

Améliorations de cette version du moteur

- Vous pouvez désormais indiquer la taille des demandes groupées ElasticSearch pour les recherches en texte intégral dans Gremlin.
- Amélioration de l'utilisation de la mémoire pour les requêtes SPARQL GROUP BY.
- Ajout d'un nouvel optimiseur de requêtes Gremlin pour affiner certains filtres indépendants.
- Augmentation de la durée maximale pendant WebSocket laquelle une connexion authentifiée via IAM peut rester ouverte, de 36 heures à 10 jours.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue selon lequel, si vous envoyiez un paramètre d'URL non encodé dans une requête POST, Neptune renvoyait le code de statut HTTP 500 et une exception `InternalServerErrorException`. Neptune renvoie maintenant le code de statut HTTP 400 et une exception `BadRequestException`, avec le message : `Failure to process the POST request parameters`.
- Correction d'un bogue de Gremlin qui empêchait le signalement correct d'un échec de WebSocket connexion.
- Correction d'un bogue Gremlin lié à la disparition des effets secondaires.
- Correction d'un bogue Gremlin qui empêchait la prise en charge correcte du paramètre de recherche en texte intégral `batchsize`.
- Correction d'un bogue Gremlin pour gérer `toV` et `fromV` individuellement pour chaque direction. `bothE`
- Correction d'un bogue Gremlin impliquant `Edge pathType` dans l'étape `hasLabel`.
- Correction d'un bogue SPARQL en raison duquel la réorganisation des jointures avec des liaisons statiques ne fonctionnait pas correctement.
- Correction d'un bogue SPARQL UPDATE LOAD en raison duquel un compartiment Amazon S3 indisponible n'était pas correctement signalé.

- Correction d'un bogue SPARQL en raison duquel un problème avec un nœud SERVICE dans une sous-requête n'était pas correctement signalé.
- Correction d'un bogue SPARQL en raison duquel les requêtes contenant des conditions FILTER EXISTS ou FILTER NOT EXISTS imbriquées n'étaient pas correctement évaluées.
- Correction d'un bogue SPARQL pour gérer correctement les liaisons générées en double lors de l'appel des points de terminaison du service SPARQL via des requêtes de génération.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.3.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.3`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.3.0

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Si le paramètre `AutoMinorVersionUpgrade` de votre cluster est défini sur `True`, votre cluster sera automatiquement mis à niveau vers cette version de moteur deux à trois semaines après la date de cette version, au cours d'une fenêtre de maintenance.

Mise à niveau vers cette version

Amazon Neptune 1.0.3.0 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
```

```
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.3.0.R3 du moteur Amazon Neptune (19/02/2021)

Depuis le 19 février 2021, la version 1.0.3.0.R3 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue dans le chargement en bloc des fichiers CSV impliquant des libellés se terminant par #.
- Correction d'un bogue Gremlin qui pouvait entraîner des résultats manquants pour un modèle spécifique de requêtes utilisant l'étape `as()`.

- Correction d'un bogue Gremlin qui pouvait provoquer des erreurs lors de l'utilisation de l'étape `project()` imbriquée dans une autre étape, comme `union()`.
- Correction d'un bogue Gremlin lié à l'exécution de traversées de chaînes dans le moteur DFE expérimental en cas d'utilisation d'une méthode de terminal comme `toList()`.
- Correction d'un bogue Gremlin qui empêchait de clôturer une transaction lors de l'utilisation de l'étape `iterate()` dans une requête sous forme de chaîne.
- Correction d'un bogue Gremlin qui pouvait provoquer une exception pour les requêtes utilisant le modèle `is(P.gte(0))` dans certaines situations.
- Correction d'un bogue Gremlin où l'exception `InternalFailureException` était définie comme code de réponse dans certaines circonstances lors d'un événement `ConcurrentModificationException`.
- Correction d'un bogue Gremlin où, dans certaines conditions, la mise à jour des arêtes ou des sommets pouvait provoquer une exception `InternalFailureException` transitoire.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.3.0.R3, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.8`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.3.0.R3

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.3.0`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.3.0.R3 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez

mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.3.0.R2 du moteur Amazon Neptune (12/10/2020)

Depuis le 12 octobre 2020, la version 1.0.3.0.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Améliorations de cette version du moteur

- Amélioration des performances pour l'étape Gremlin `properties()`.
- Ajout de détails sur `BindOp` et `MultiplexerOp` dans les rapports d'explication et de profil.
- Pour les réponses aux requêtes SPARQL, `charset` a été ajouté à l'en-tête Content-Type, permettant aux clients HTTP de reconnaître le jeu de caractères utilisé automatiquement.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue SPARQL qui empêchait la gestion de l'exception `CancellationException`
- Correction d'un bogue SPARQL qui empêchait les requêtes contenant des options imbriquées de fonctionner correctement.
- Correction d'un bogue SPARQL dans LOAD où `ConcurrentModificationException` pouvait provoquer le blocage d'une requête.
- Correction d'un bogue SPARQL qui empêchait les réponses aux requêtes d'être compressées au format gzip.
- Correction d'un bogue Gremlin dans l'étape `groupBy()`.
- Correction d'un bogue Gremlin lié à l'utilisation d'une étape `aggregate()` au sein d'une étape `local()`.
- Correction d'un bogue Gremlin lié à l'utilisation de `bothE()` suivi d'un prédicat utilisant des valeurs agrégées.
- Correction d'un bogue Gremlin lié à l'utilisation de l'étape `bothE()` au sein d'une étape `repeat()`.
- Correction d'une fuite de mémoire Grenlin potentielle liée à l'étape `both()`.
- Correction d'un bogue en raison duquel les métriques des demandes manquaient parce qu'un point de terminaison se terminant par / n'était pas géré correctement.
- Correction d'un bogue qui pouvait générer une exception `ThrottlingException` même si la file d'attente des requêtes n'était pas pleine.

- Correction d'un bogue lié à la récupération de l'état du chargement lorsqu'un chargement échoue pour une raison telle que `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE`.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.3.0.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.3`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.3.0.R2

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Si le paramètre `AutoMinorVersionUpgrade` de votre cluster est défini sur `True`, votre cluster sera automatiquement mis à niveau vers cette version de moteur deux à trois semaines après la date de cette version, au cours d'une fenêtre de maintenance.

Mise à niveau vers cette version

Amazon Neptune 1.0.3.0.R2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un

instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.2 du moteur Amazon Neptune (09/03/2020)

Depuis le 9 mars 2020, la version 1.0.2.2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Versions de correctifs ultérieures pour cette version

- Sortie : 1.0.2.2.R2 (02/04/2020)
- Sortie : 1.0.2.2.R3 (22/07/2020)
- Sortie : 1.0.2.2.R4 (23/07/2020)
- Sortie : 1.0.2.2.R5 (12/10/2020)
- Sortie : 1.0.2.2.R6 (19/02/2021)

Améliorations de cette version du moteur

- Ajout d'informations à l'API d'état sur les transactions qui sont annulées. Consultez Statut d'une instance.
- Mise à niveau de la version d'Apache TinkerPop vers la version 3.4.3.

  La version 3.4.3 est rétrocompatible avec la version précédente prise en charge par Neptune (3.4.1). Elle introduit un changement mineur dans le comportement : Gremlin ne renvoie plus d'erreur lorsque vous essayez de fermer une session qui n'existe pas (consultez les informations relatives à la prévention des erreurs lors de la clôture de sessions qui n'existent pas).
- Suppression des goulets d'étranglement des performances dans l'exécution des étapes de recherche en texte intégral de Gremlin.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue SPARQL dans la gestion des modèles de graphique vide dans les requêtes.
- Correction d'un bogue SPARQL dans la gestion des points-virgules non codés dans les requêtes codées en URL.
- Correction d'un bogue Gremlin dans la gestion des sommets répétés dans l'étape `Union`.
- Correction d'un bogue Gremlin qui faisait que certaines requêtes avec un `.simplePath()` ou `.cyclicPath()` dans un `.repeat()` renvoyaient des résultats incorrects.
- Correction d'un bogue Gremlin qui faisant que `.project()` renvoyait des résultats incorrects si sa traversée enfant ne renvoyait aucune solution.
- Correction d'un bogue Gremlin où les erreurs provenant de conflits de lecture-écriture déclenchaient un `InternalFailureException` plutôt qu'un `ConcurrentModificationException`.
- Correction d'un bogue Gremlin qui provoquait des échecs `.group().by(...).by(values("property"))`.
- Correction de bogues Gremlin dans la sortie du profil pour les full-text-search étapes.
- Correction d'une fuite de ressource dans les sessions Gremlin.
- Correction d'un bogue qui empêchait l'API d'état de signaler la bonne version à commander dans certains cas.
- Correction d'un bogue lié au chargeur en bloc qui autorisait qu'une URL à un emplacement autre qu'Amazon S3 soit utilisée comme source dans une demande de chargement en bloc.

- Correction d'un bogue lié au chargeur en bloc dans le statut détaillé du chargement.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.2.2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.3`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.2.2

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Si le paramètre `AutoMinorVersionUpgrade` de votre cluster est défini sur `True`, votre cluster sera automatiquement mis à niveau vers cette version de moteur deux à trois semaines après la date de cette version, au cours d'une fenêtre de maintenance.

Mise à niveau vers cette version

Amazon Neptune 1.0.2.2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.2.R6 du moteur Amazon Neptune (19/02/2021)

Depuis le 19 février 2021, la version 1.0.2.2.R6 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin où l'exception `InternalFailureException` était définie comme code de réponse dans certaines circonstances lors d'un événement `ConcurrentModificationException`.
- Correction d'un bogue Gremlin où, dans certaines conditions, la mise à jour des arêtes ou des sommets pouvait provoquer une exception `InternalFailureException` transitoire.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.2.2.R6, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.8`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.2.2.R6

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.2.2`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.2.2.R6 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
> ```

```
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.2.R5 du moteur Amazon Neptune (12/10/2020)

Depuis le 12 octobre 2020, la version 1.0.2.2.R5 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Améliorations de cette version du moteur

- Amélioration des performances pour l'étape Gremlin `properties()`.
- Ajout de détails sur `BindOp` et `MultiplexerOp` dans les rapports d'explication et de profil.
- Pour les réponses aux requêtes SPARQL, `charset` a été ajouté à l'en-tête Content-Type, permettant aux clients HTTP de reconnaître le jeu de caractères utilisé automatiquement.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue SPARQL qui empêchait la gestion de l'exception `CancellationException`
- Correction d'un bogue SPARQL qui empêchait les requêtes contenant des options imbriquées de fonctionner correctement.
- Correction d'un bogue SPARQL dans LOAD où `ConcurrentModificationException` pouvait provoquer le blocage d'une requête.
- Correction d'un bogue SPARQL qui empêchait les réponses aux requêtes d'être compressées au format gzip.
- Correction d'un bogue Gremlin dans l'étape `groupBy()`.

- Correction d'un bogue Gremlin lié à l'utilisation d'une étape `aggregate()` au sein d'une étape `local()`.
- Correction d'un bogue Gremlin lié à l'utilisation de `bothE()` suivi d'un prédicat utilisant des valeurs agrégées.
- Correction d'un bogue Gremlin lié à l'utilisation de l'étape `bothE()` au sein d'une étape `repeat()`.
- Correction d'une fuite de mémoire Grenlin potentielle liée à l'étape `both()`.
- Correction d'un bogue en raison duquel les métriques des demandes manquaient parce qu'un point de terminaison se terminant par / n'était pas géré correctement.
- Correction d'un bogue qui pouvait générer une exception `ThrottlingException` même si la file d'attente des requêtes n'était pas pleine.
- Correction d'un bogue lié à la récupération de l'état du chargement lorsqu'un chargement échoue pour une raison telle que `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE`.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.2.2.R5, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.3`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.2.2.R5

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.2.2`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.2.2.R5 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données

sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.2.R4 du moteur Amazon Neptune (23/07/2020)

Depuis le 23 juillet 2020, la version 1.0.2.2.R4 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Améliorations de cette version du moteur

- Amélioration de l'utilisation de la mémoire en libérant plus fréquemment la mémoire inutilisée dans le système d'exploitation.
- L'utilisation de la mémoire a également été améliorée pour les requêtes SPARQL GROUP BY.
- La durée maximale pendant laquelle une WebSocket connexion authentifiée par IAM peut rester ouverte est passée de 36 heures à 10 jours.
- Ajout de la `BufferCacheHitRatio` CloudWatch métrique, qui peut être utile pour diagnostiquer la latence des requêtes et ajuster les types d'instances. Consultez Métriques Neptune.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue lors de la fermeture des WebSocket connexions IAM inactives ou expirées. Neptune envoie désormais un frame de fermeture avant de fermer la connexion.
- Correction d'un bogue SPARQL dans l'évaluation des requêtes contenant des conditions FILTER EXISTS FILTER NOT and/or EXISTS imbriquées.
- Correction d'un bogue de résiliation des requêtes SPARQL qui bloquait les threads sur le serveur dans certaines conditions extrêmes.
- Correction d'un bogue Gremlin impliquant Edge PathType dans l'étape `hasLabel`.
- Correction d'un bogue Gremlin pour gérer `toV` et `fromV` individuellement pour chaque direction. `bothE`
- Correction d'un bogue Gremlin lié à la disparition des effets secondaires.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.2.2.R4, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.3`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.2.2.R4

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.2.2`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.2.2.R4 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise

à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the instance before
>   proceeding with the upgrade.
> ```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.2.R3 du moteur Amazon Neptune (22/07/2020)

La version du moteur 1.0.2.2.R3 a été intégrée à la version du moteur 1.0.2.2.R4.

Version 1.0.2.2.R2 du moteur Amazon Neptune (02/04/2020)

Depuis le 2 avril 2020, la version 1.0.2.2.R2 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Améliorations de cette version du moteur

- Vous pouvez maintenant mettre en file d'attente jusqu'à 64 tâches de chargement en bloc, au lieu d'attendre la fin d'une tâche avant de lancer la suivante. Vous pouvez également subordonner l'exécution d'une demande de chargement en file d'attente à la réussite d'une ou de plusieurs tâches de chargement précédemment en file d'attente à l'aide du paramètre `dependencies` de la commande `load`. Consultez Commande de chargeur Neptune.
- Full-text-search la sortie peut maintenant être triée (voirParamètres de recherche en texte intégral).
- Il existe maintenant un paramètre de cluster de bases de données permettant d'appeler des flux Neptune. Cette fonctionnalité n'est plus en mode expérimental. Consultez Activation de Neptune Streams.

Défauts corrigés dans cette version du moteur

- Correction d'une défaillance aléatoire au démarrage du serveur qui retardait la création de l'instance.

- Correction d'un problème d'optimiseur dans le cadre duquel les instructions `BIND` de la requête faisaient démarrer l'optimiseur avec des motifs non sélectifs dans la planification des ordres de jointure.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.2.2.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.3`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.2.2.R2

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.2.2`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.2.2.R2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.1 du moteur Amazon Neptune (22/11/2019)

Versions de correctifs ultérieures pour cette version

- Sortie : 1.0.2.1.R6 (22/04/2020)
- Sortie : 1.0.2.1.R5 (22/04/2020) Cette version de correctif n'a pas été déployée.
- Sortie : 1.0.2.1.R4 (20/12/2019)
- Sortie : 1.0.2.1.R3 (12/12/2019)
- Sortie : 1.0.2.1.R2 (25/11/2019)

Nouvelles fonctionnalités pour cette version du moteur

- Ajout de fonctionnalités de recherche en texte intégral grâce à l'intégration avec Amazon OpenSearch Service. Consultez Recherche en texte intégral Neptune.

- Ajout de l'option d'utilisation du mode Lab afin de créer un quatrième index (un index OSGP) pour les nombres élevés de prédicats. Consultez Index OSGP.
- Ajout d'un mode détaillé à SPARQL Explain. Consultez Utilisation de SPARQL `explain` et Sortie de mode détaillé pour en savoir plus.
- Ajout d'informations sur le mode Lab au rapport de statut du moteur. Consultez Statut d'une instance pour plus de détails.
- Les instantanés du cluster de bases de données peuvent désormais être copiés d'une AWS région à l'autre. Consultez Copie d'un instantané.

Améliorations de cette version du moteur

- Amélioration des performances lors du traitement d'un grand nombre de prédicats.
- Optimisation renforcée des requêtes. Même si cela devrait être entièrement transparent pour les clients, nous vous encourageons à tester vos applications avant de les mettre à niveau afin de vous assurer qu'elles se comportent comme prévu.
- Améliorations mineures pour la génération de rapport d'erreurs.
- Ajout d'optimisations pour les étapes Gremlin `.project()` et `.identity()`.
- Ajout d'optimisations pour les cas `.union()` Gremlin non terminaux.
- Ajout d'une prise en charge native pour les traversées `.path().by()` de Gremlin.
- Ajout d'une prise en charge native pour `.coalesce()` de Gremlin.
- Optimisation supplémentaire de l'écriture en bloc.
- Nous exigeons désormais que les connexions HTTPS utilisent au moins la version TLS 1.2 ou supérieure, afin d'empêcher l'utilisation de outdated/insecure chiffrements.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue de traitement de traversée interne `addE()` de Gremlin.
- Correction d'un bogue Gremlin provoqué par la fuite d'annotations AST de traversées enfants vers le parent.
- Correction d'un bogue qui avait lieu dans Gremlin quand `.otherV()` était appelé après `select()`.
- Correction d'un bogue Gremlin qui provoquait l'échec de certaines étapes `.hasLabel()` si elles apparaissaient après une étape `bothE()`.

- Corrections mineures pour .sum() and .project() dans Gremlin.
- Correction d'un bogue dans le traitement des requêtes SPARQL sans accolade de fermeture.
- Correction de quelques bogues mineurs dans SPARQL Explain.
- Correction d'un bogue dans le traitement des demandes d'obtention de statut de chargement simultanées.
- Mémoire réduite utilisée pour exécuter certaines traversées Gremlin avec des étapes `.project()`.
- Correction apportée aux comparaisons numériques de valeurs spéciales dans SPARQL. Consultez Conformité aux normes.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.2.1, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.1`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.2.1

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

La mise à niveau vers cette version n'est pas automatique.

Mise à niveau vers cette version

Amazon Neptune 1.0.2.1 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.1.R6 du moteur Amazon Neptune (22/04/2020)

Depuis le 22 avril 2020, la version 1.0.2.1.R6 du moteur est déployée globalement. Notez que plusieurs jours sont nécessaires pour qu'une nouvelle version soit disponible dans chaque région.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue au cours duquel `ConcurrentModificationConflictException` et `TransactionException` n'étaient pas convertis en un élément `NeptuneGremlinException`, provoquant le renvoi de `InternalFailureException` aux clients.
- Correction d'un bogue où Neptune signalait que son état était sain avant que le serveur ne soit complètement prêt.
- Correction d'un bogue au cours duquel les validations des transactions du dictionnaire et de l'utilisateur étaient hors service lorsque deux mappages `value->id` étaient insérés simultanément.
- Correction d'un bogue dans la sérialisation de l'état de charge.
- Correction d'un bogue des sessions Gremlin.
- Correction d'un bogue où Neptune ne parvenait pas à émettre une exception lorsque le serveur ne démarrait pas.
- Correction d'un bogue où Neptune ne parvenait pas à envoyer un frame de fermeture WebSocket avant de fermer le canal.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.2.1.R6, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.1`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.2.1.R6

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.2.1`.

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.2.1.R6 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.1.R5 du moteur Amazon Neptune (22/04/2020)

La version 1.0.2.1.R5 du moteur n'a jamais été déployée.

Version 1.0.2.1.R4 du moteur Amazon Neptune (20/12/2019)

Améliorations de cette version du moteur

- Neptune essaie désormais de toujours placer un full-text-search appel en premier dans le pipeline d'exécution. Cela réduit le volume des appels OpenSearch, ce qui peut améliorer considérablement les performances. Consultez Full-text-search exécution de requêtes.
- Neptune génère désormais une exception `IllegalArgumentException` si vous essayez d'accéder à une propriété, un sommet ou une arête qui n'existe pas. Auparavant, Neptune générait une exception `UnsupportedOperationException` dans cette situation.

  Par exemple, si vous essayez d'ajouter une arête référençant un sommet inexistant, vous générerez désormais une `IllegalArgumentException`.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin dans lequel une traversée `union` à l'intérieur d'un `project-by` ne renvoie aucun résultat ou renvoie des résultats incorrects.
- Correction d'un bogue Gremlin qui faisait renvoyer des résultats incorrects aux étapes `.project().by()` imbriquées.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.2.1.R4, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.1`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.2.1.R4

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Toutefois, la mise à jour automatique vers cette version n'est pas prise en charge.

Mise à niveau vers cette version

Amazon Neptune 1.0.2.1.R4 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations,

l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.1.R3 du moteur Amazon Neptune (12/12/2019)

Défauts corrigés dans cette version du moteur

- Correction d'un bogue qui entraînait la désactivation de l'index `OSGP`, même si la fonctionnalité était correctement dansMode Lab à l'aide de la valeur `ObjectIndex` dans le paramètre `neptune_lab_mode`.
- Correction d'un bogue qui affectait les requêtes Gremlin avec un `.fold()` dans une étape `.project().by()`. Par exemple, la requête suivante a renvoyé des résultats incomplets :

```
g.V().project("a").by(valueMap().fold())
```

- Correction d'un goulot d'étranglement des performances dans les chargements en groupe de données RDF.
- Correction d'un bogue qui provoquait un incident sur les réplicas lorsque les flux étaient activés et que le réplica était redémarré avant l'instance principale.
- Correction d'un bogue où les certificats SSL pivotés sur les instances n'étaient pas récupérés sans un redémarrage de l'instance.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.2.1.R3, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.1`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.2.1.R3

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Toutefois, la mise à jour automatique vers cette version n'est pas prise en charge.

Mise à niveau vers cette version

Amazon Neptune 1.0.2.1.R3 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations,

l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.1.R2 du moteur Amazon Neptune (25/11/2019)

Défauts corrigés dans cette version du moteur

- Correction d'un bogue affectant toutes les requêtes `project().by()` avec les traversées de type .by autres que rond-robin (tourniquet) et `path()`.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.2.1.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.1`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.2.1.R2

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Toutefois, la mise à jour automatique vers cette version n'est pas prise en charge.

Mise à niveau vers cette version

Amazon Neptune 1.0.2.1.R2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
```

```
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.0 du moteur Amazon Neptune (08/11/2019)

IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE

À partir du 19 mai 2020, plus aucune instance ne sera créée avec cette version du moteur.

Cette version du moteur est désormais remplacée par la version 1.0.2.1, qui contient toutes les corrections de bogues de cette version ainsi que des fonctionnalités supplémentaires telles que

l'intégration de la recherche en texte intégral, la prise en charge des index OSGP et la copie de clusters de instantanés de base de données entre les régions. AWS

À compter du 1er juin 2020, Neptune mettra automatiquement à niveau tout cluster exécutant cette version de moteur vers le dernier correctif de la version 1.0.2.1 lors de la prochaine fenêtre de maintenance. Vous pouvez effectuer une mise à niveau manuelle avant cette date, comme décrit ici.

Si vous rencontrez des problèmes avec la mise à niveau, contactez-nous via AWS Support ou les Forums des développeurs AWS.

Versions de correctifs ultérieures pour cette version

- Sortie : 1.0.2.0.R3 (05/05/2020)
- Sortie : 1.0.2.0.R2 (21/11/2019)

Nouvelles fonctionnalités pour cette version du moteur

Outre les mises à jour de maintenance, cette version ajoute de nouvelles fonctionnalités pour prendre en charge plusieurs versions de moteur à la fois (voir Maintenance du cluster de bases de données Amazon Neptune).

Par conséquent, la numérotation des version du moteur a changé (voir Numérotation des versions avant la version du moteur 1.3.0.0).

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.2.0, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.1`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.2.0

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

La mise à niveau vers cette version n'est pas automatique.

Mise à niveau vers cette version

Amazon Neptune 1.0.2.0 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations,

l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.0.R3 du moteur Amazon Neptune (05/05/2020)

IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE

À partir du 19 mai 2020, plus aucune instance ne sera créée avec cette version du moteur.

Cette version du moteur est désormais remplacée par la version 1.0.2.1, qui contient toutes les corrections de bogues de cette version ainsi que des fonctionnalités supplémentaires telles que l'intégration de la recherche en texte intégral, la prise en charge des index OSGP et la copie de clusters de instantanés de base de données entre les régions. AWS

À compter du 1er juin 2020, Neptune mettra automatiquement à niveau tout cluster exécutant cette version de moteur vers le dernier correctif de la version 1.0.2.1 lors de la prochaine fenêtre de maintenance. Vous pouvez effectuer une mise à niveau manuelle avant cette date, comme décrit ici.

Si vous rencontrez des problèmes avec la mise à niveau, contactez-nous via AWS Support ou les Forums des développeurs AWS.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue au cours duquel `ConcurrentModificationConflictException` et `TransactionException` étaient signalés en tant qu'éléments `InternalFailureException` génériques.
- Correction de bogues lors des vérifications de l'état qui provoquaient des redémarrages fréquents du serveur lors du démarrage.
- Correction d'un bogue au cours duquel les données n'étaient pas visibles sur les réplicas parce que les validations n'étaient pas fonctionnelles dans certaines conditions.
- Correction d'un bogue lié à la sérialisation de l'état de charge au cours duquel une charge échouait en raison d'un manque d'autorisations d'Amazon S3.
- Correction d'une fuite de ressource dans les sessions Gremlin.
- Correction d'un bogue lié à la surveillance de l'état qui masquait l'état défectueux au démarrage des composants gérant l'authentification IAM.
- Correction d'un bug qui empêchait Neptune d'envoyer un WebSocket cadre de fermeture avant de fermer le canal.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.2.0.R3, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.1`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.2.0.R3

Votre cluster sera automatiquement mis à niveau vers cette version de correctif lors de votre prochaine fenêtre de maintenance si vous exécutez la version de moteur `1.0.2.0`.

Vous pouvez mettre à niveau manuellement n'importe quelle version précédente du moteur Neptune vers cette version.

Mise à niveau vers cette version

Amazon Neptune 1.0.2.0.R3 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.2.0.R2 du moteur Amazon Neptune (21/11/2019)

IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE

À partir du 19 mai 2020, plus aucune instance ne sera créée avec cette version du moteur.

Cette version du moteur est désormais remplacée par la version 1.0.2.1, qui contient toutes les corrections de bogues de cette version ainsi que des fonctionnalités supplémentaires telles que l'intégration de la recherche en texte intégral, la prise en charge des index OSGP et la copie de clusters de instantanés de base de données entre les régions. AWS

À compter du 1er juin 2020, Neptune mettra automatiquement à niveau tout cluster exécutant cette version de moteur vers le dernier correctif de la version 1.0.2.1 lors de la prochaine fenêtre de maintenance. Vous pouvez effectuer une mise à niveau manuelle avant cette date, comme décrit ici.

Si vous rencontrez des problèmes avec la mise à niveau, contactez-nous via AWS Support ou les Forums des développeurs AWS.

Défauts corrigés dans cette version du moteur

- Amélioration de la stratégie de mise en cache pour les pages à valider sur le serveur pour une récupération plus rapide de `FreeableMemory` lorsque le serveur passe à un état de faible mémoire.
- Correction d'un bogue qui pouvait provoquer une course et un crash lorsque de nombreuses demandes simultanées de chargement initial étaient traitées sur le serveur. and/or

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.2.0.R2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.1`
- Version SPARQL : `1.1`

Chemins de mise à niveau vers la version de moteur 1.0.2.0.R2

Vous pouvez mettre à niveau manuellement n'importe quelle version antérieure du moteur Neptune vers cette version.

Toutefois, la mise à jour automatique vers cette version n'est pas prise en charge.

Mise à niveau vers cette version

Amazon Neptune 1.0.2.0.R2 est désormais disponible globalement.

Si un cluster de bases de données exécute une version de moteur à partir de laquelle il existe un chemin de mise à niveau vers cette version, il peut être mis à niveau dès maintenant. Vous pouvez mettre à niveau n'importe quel cluster éligible à l'aide des opérations de cluster de bases de données sur la console ou à l'aide du kit SDK. La commande CLI suivante met immédiatement à niveau un cluster éligible :

Pour Linux, OS X ou Unix :

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Pour Windows :

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur ces instances. Vous subirez donc un temps d'arrêt allant de 20-30 secondes à plusieurs minutes, après quoi vous pourrez reprendre l'utilisation du cluster de bases de données.

Toujour effectuer des tests avant la mise à niveau

Lorsqu'une nouvelle version majeure ou mineure du moteur Neptune est publiée, testez toujours vos applications Neptune sur cette version avant de procéder à la mise à niveau. Même une mise à niveau mineure peut introduire de nouvelles fonctionnalités ou de nouveaux comportements susceptibles d'affecter le code.

Commencez par comparer les pages de notes de mise à jour de votre version actuelle à celles de la version cible pour déterminer s'il existe des modifications des versions de langage de requête ou d'autres changements majeurs.

Le meilleur moyen de tester une nouvelle version avant de mettre à niveau le cluster de bases de données de production est de cloner ce cluster pour qu'il exécute cette nouvelle version du moteur. Vous pouvez ainsi exécuter des requêtes sur le clone sans affecter le cluster de bases de données de production.

Toujours créer un instantané manuel avant de procéder à la mise à niveau

Avant la mise à niveau, nous vous recommandons vivement de toujours créer un instantané manuel du cluster de bases de données. Un instantané automatique n'offre qu'une protection à court terme, tandis qu'un instantané manuel reste disponible jusqu'à ce que vous le supprimiez explicitement.

Dans certains cas, Neptune crée un instantané manuel pour vous dans le cadre du processus de mise à niveau, mais il est préférable de ne pas compter sur ce mécanisme et de créer dans tous les cas votre propre instantané manuel.

Lorsque vous êtes certain de ne pas avoir besoin de rétablir l'état antérieur à la mise à niveau de votre cluster de bases de données, vous pouvez supprimer explicitement l'instantané manuel que

vous avez créé vous-même, ainsi que celui que Neptune a éventuellement créé. Si Neptune crée un instantané manuel, il porte un nom commençant par `preupgrade`, suivi du nom de votre cluster de bases de données, de la version du moteur source, de la version du moteur cible et de la date.

> **Note**
>
> Si vous essayez de procéder à une mise à niveau alors qu'une action en attente est en cours, une erreur telle que la suivante peut survenir :
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si vous rencontrez cette erreur, attendez que l'action en attente soit terminée ou déclenchez immédiatement une fenêtre de maintenance pour laisser la mise à niveau précédente se terminer.

Pour plus d'informations sur la mise à niveau de la version du moteur , consultez Maintenance du cluster de bases de données Amazon Neptune. Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Version 1.0.1.2 du moteur Amazon Neptune (10/06/2020)

IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE

À partir du 27 avril 2021, plus aucune instance ne sera créée avec cette version du moteur.

Améliorations de cette version du moteur

- Neptune génère désormais une exception `IllegalArgumentException` si vous essayez d'accéder à une propriété, un sommet ou une arête qui n'existe pas. Auparavant, Neptune générait une exception `UnsupportedOperationException` dans cette situation.

Par exemple, si vous essayez d'ajouter une arête référençant un sommet inexistant, vous générerez désormais une `IllegalArgumentException`.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue au cours duquel les validations des transactions du dictionnaire et de l'utilisateur étaient hors service lorsque deux mappages `value->id` étaient insérés simultanément.
- Correction d'un bogue dans la sérialisation de l'état de charge.
- Correction d'une défaillance aléatoire au démarrage du serveur qui retardait la création de l'instance.
- Correction d'une fuite de curseur.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.1.2, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.4.1`
- Version SPARQL : `1.1`

Version 1.0.1.1 du moteur Amazon Neptune (26/06/2020)

IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE

À partir du 27 avril 2021, plus aucune instance ne sera créée avec cette version du moteur.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue où les validations n'étaient pas fonctionnelles lorsqu'elles étaient insérées simultanément.
- Correction d'un bogue dans la sérialisation de l'état de charge.

- Correction d'une défaillance aléatoire au démarrage du serveur qui retardait la création de l'instance.
- Correction d'une fuite de mémoire.

Versions de langage de requête prises en charge dans cette version

Avant de mettre à niveau un cluster de bases de données vers la version 1.0.1.1, assurez-vous que votre projet est compatible avec les versions de langage de requête suivantes :

- Version Gremlin : `3.3.2`
- Version SPARQL : `1.1`

Version 1.0.1.0 du moteur Amazon Neptune (02/07/2019)

IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE

À partir du 27 avril 2021, plus aucune instance ne sera créée avec cette version du moteur.

Mises à jour du moteur Amazon Neptune 31/10/2019

Version : 1.0.1.0.200502.0

IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE

À partir du 27 avril 2021, plus aucune instance ne sera créée avec cette version du moteur.

Défauts corrigés dans cette version du moteur

- Correction d'un bug Gremlin dans la sérialisation de la réponse de l'étape `tree()` lorsque des clients se connectent à Neptune en utilisant `traversal().withRemote(...)` (en d'autres termes, à l'aide du bytecode GLV).

  Cette version résout un problème dans lequel les clients qui se connectaient à Neptune à l'aide d'un `traversal().withRemote(...)` recevait une réponse non valide aux requêtes Gremlin contenant une étape `tree()`.

- Correction d'un bogue SPARQL dans les requêtes `DELETE WHERE LIMIT`, dans lequel le processus de fin de requête se bloquait en raison d'une condition de concurrence, provoquant l'expiration de la requête.

Mises à jour du moteur Amazon Neptune 15/10/2019

Version : 1.0.1.0.200463.0

IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE

À partir du 27 avril 2021, plus aucune instance ne sera créée avec cette version du moteur.

Nouvelles fonctionnalités pour cette version du moteur

- Ajout d'une Explain/Profile fonctionnalité Gremlin (voir`Analyse de l'exécution des requêtes Neptune à l'aide de Gremlin (explication)`).
- Ajout de Prise en charge des sessions basées sur des scripts Gremlin pour activer l'exécution de plusieurs traversées Gremlin dans une seule transaction.
- Ajout de la prise en charge de l'extension de requête fédérée SPARQL dans Neptune (voir Requête fédérée SPARQL 1.1 et `Requêtes fédérées SPARQL dans Neptune à l'aide de l'extension SERVICE`).
- Ajout d'une fonctionnalité vous permettant d'injecter votre propre `queryId` dans une requête Gremlin ou SPARQL, via un paramètre d'URL HTTP ou via un indicateur de requête `queryId` SPARQL (voir Injection d'un ID personnalisé dans une requête Neptune Gremlin ou SPARQL).
- Ajout d'une fonctionnalité Mode Lab à Neptune qui vous permet de tester les fonctionnalités à venir qui ne sont pas encore prêtes à être utilisées en production.
- Ajout d'une fonctionnalité Flux Neptune à venir qui consigne de manière fiable chaque modification apportée à votre base de données dans un flux conservé pendant une semaine. Cette fonction est disponible uniquement en mode Lab.
- Mise à jour de la sémantique formalisée pour les transactions simultanées (voir Sémantique des transactions dans Neptune). Cette fonctionnalité offre des garanties de conformité aux normes du secteur en matière de simultanéité.

  Cette sémantique des transactions personnalisée est activée par défaut. Dans certains scénarios, cette fonctionnalité peut modifier le comportement de chargement actuel et réduire les performances de chargement. Vous pouvez utiliser le paramètre `neptune_lab_mode`

de cluster de bases de données pour revenir à la sémantique précédente en incluant `ReadWriteConflictDetection=disabled` dans la valeur du paramètre.

Améliorations de cette version du moteur

- Amélioration de l'Statut d'une instanceAPI en indiquant la version TinkerPop et la version de SPARQL utilisées par le moteur.
- Amélioration des performances de l'opérateur de sous-graphe Gremlin.
- Amélioration des performances de la sérialisation des réponses Gremlin.
- Amélioration des performances à l'étape Gremlin Union.
- Amélioration de la latence des requêtes SPARQL simples.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue Gremlin pour lequel le délai d'attente était renvoyé de manière incorrecte en tant qu'échec interne.
- Correction d'un bogue SPARQL dans lequel le paramètre ORDER BY exécuté sur un ensemble partiel de variables provoquait une erreur de serveur interne.

Mises à jour du moteur Amazon Neptune 19/09/2019

IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE

À partir du 27 avril 2021, plus aucune instance ne sera créée avec cette version du moteur.

Version : 1.0.1.0.200457.0

Amazon Neptune 1.0.1.0.200457.0 est disponible globalement. Tous les nouveaux clusters de bases de données Neptune, y compris ceux restaurés à partir d'instantanés, seront créés dans 1.0.1.0.200457.0 une fois la mise à jour du moteur terminée pour cette région.

Les clusters existants peuvent être mis immédiatement à niveau vers cette version à l'aide des opérations de base de données sur la console ou à l'aide du kit SDK. Vous pouvez utiliser la commande CLI suivante pour mettre à niveau un cluster de bases de données :

```
aws neptune apply-pending-maintenance-action \
```

```
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur toutes les instances figurant dans un cluster de bases de données, si bien que vous connaîtrez de 20-30 secondes à plusieurs minutes d'indisponibilité, après quoi vous pourrez reprendre l'utilisation de votre ou de vos clusters de bases de données. Vous pouvez consulter et modifier vos paramètres de fenêtre de maintenance dans la console Neptune.

Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Défauts corrigés dans cette version du moteur

- Correction d'un problème d'exactitude Gremlin introduit dans la version précédente du moteur (1.0.1.0.200369.0) en supprimant l'amélioration des performances de la gestion des prédicats cumulés qui l'entraînait.
- Correction d'un bogue SPARQL qui provoquait la génération d'une `InternalServerError` par les requêtes avec le paramètre `DISTINCT` et un modèle unique encapsulé dans `OPTIONAL`.

Mises à jour du moteur Amazon Neptune 13/08/2019

IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE

À partir du 27 avril 2021, plus aucune instance ne sera créée avec cette version du moteur.

Nouvelles fonctionnalités pour cette version du moteur

- Ajout d'une option `OVERSUBSCRIBE` au paramètre `parallelism` de Commande de chargeur Neptune, ce qui conduit le chargeur en bloc Neptune à utiliser tous les threads et ressources disponibles.

Améliorations de cette version du moteur

- Amélioration des performances des filtres SPARQL contenant des expressions OR logiques simples.

- Amélioration des performances de Gremlin dans la gestion des prédicats conjonctifs.

Défauts corrigés dans cette version du moteur

- Correction d'un bogue SPARQL empêchant la soustraction d'un élément `xsd:duration` à un élément `xsd:date`.
- Correction d'un bogue SPARQL entraînant des résultats incomplets de l'inlining statique en présence d'un `UNION`.
- Correction d'un bogue SPARQL dans l'annulation de requête.
- Correction d'un bogue Gremlin entraînant un dépassement de capacité au cours de la promotion de type.
- Correction d'un bogue Gremlin dans la gestion des éléments de sommet dans les étapes `addE().from().to()`.
- Correction d'un bogue Gremlin (publié le 26/07/2019 dans la version 1.0.1.0 200366.0 du moteur) impliquant la gestion des valeurs flottantes et des doubles NaN dans les insertions à cardinalité unique.
- Correction d'un bogue dans la génération de plans de requête impliquant des recherches basées sur les propriétés.

Mises à jour du moteur Amazon Neptune 26/07/2019

Version : 1.0.1.0.200366.0

IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE

À partir du 27 avril 2021, plus aucune instance ne sera créée avec cette version du moteur.

Nouvelles fonctionnalités pour cette version du moteur

- Mise à niveau vers la TinkerPop version 3.4.1 (voir Informations de TinkerPop mise à niveau et journal des modifications TinkerPop 3.4.1).

  Ces modifications offrent aux utilisateurs de Neptune de nouvelles fonctionnalités et des améliorations, notamment les suivantes :

  - `GraphBinary` est désormais disponible dans un format de sérialisation.

- Un bogue de maintien en vie qui provoquait des fuites de mémoire dans le pilote TinkerPop Java a été corrigé. Il n'est donc plus nécessaire de le contourner.

Toutefois, dans quelques cas, elles peuvent affecter le code Gremlin existant dans Neptune. Par exemple :

- `valueMap()` renvoie désormais un `Map<Object,Object>` au lieu d'un `Map<String,Object>`.
- Le comportement incohérent de l'étape `within()` a été corrigé pour la rendre compatible avec les autres étapes. Auparavant, les types devaient correspondre pour que les comparaisons fonctionnent. Désormais, il est possible de comparer des nombres de types différents avec précision. Par exemple, `33` est maintenant considéré comme égal à `33L`, ce qui n'était pas le cas auparavant.
- Un bogue dans `ReducingBarrierStep` a été corrigé, si bien qu'il ne renvoie plus de valeur si aucun élément n'est disponible pour la sortie.
- L'ordre des portées `select()` a changé (l'ordre est désormais `maps`, `side-effects`, `paths`). Cela a pour effet de changer les résultats des rares requêtes qui combinent `side-effects` et `select` avec le même nom de clé pour `side-effects` que pour `select`.
- `bulkSet()` fait désormais partie du protocole GraphSON. Les requêtes qui se terminent par `toBulkSet()` ne fonctionnent pas avec les anciens clients.
- Un paramétrage de l'étape `Submit()` a été supprimée du client 3.4.

De nombreuses autres modifications introduites dans la version TinkerPop 3.4 n'affectent pas le comportement actuel de Neptune. Par exemple, `io()` de Gremlin a été ajouté en tant qu'étape à `Traversal` et est désormais obsolète dans `Graph`, mais n'a jamais été activé dans Neptune.

- Ajout de la prise en charge des propriétés de sommet à cardinalité unique au chargeur en bloc pour Gremlin pour charger les données de graphes de propriétés.
- Ajout d'une option permettant de remplacer les valeurs existantes d'une propriété à cardinalité unique dans le chargeur en bloc.
- Ajout de la possibilité de récupérer le statut d'une requête Gremlin et d'annuler une requête Gremlin.
- Ajout d'un indicateur de requête pour les délais d'expiration de requête SPARQL.
- Ajout de la possibilité d'afficher le rôle d'instance dans l'API de statut (consultez Statut d'une instance).

- Ajout de la prise en charge du clonage de base de données (consultez Clonage de bases de données dans Neptune).

Améliorations de cette version du moteur

- Amélioration de l'explication des requêtes SPARQL pour afficher les variables de graphe à partir des clauses FROM.
- Amélioration des performances de SPARQL dans les filtres, les filtres « égal à », les clauses VALUES et les nombres de plages.
- Amélioration des performances pour le classement des étapes Gremlin.
- Amélioration des performances pour les traversées `.repeat.dedup` Gremlin.
- Amélioration des performances pour les traversées `valueMap()` et `path().by()` Gremlin.

Défauts corrigés dans cette version du moteur

- Résolution de plusieurs problèmes liés aux chemins de propriété SPARQL, y compris à l'opération avec des graphes nommés.
- Résolution d'un problème lié aux requêtes SPARQL CONSTRUCT entraînant des problèmes de mémoire.
- Résolution d'un problème lié à l'analyseur RDF Turtle et aux noms locaux.
- Résolution d'un problème de correction des messages d'erreur affichés aux utilisateurs.
- Résolution d'un problème lié aux traversées `repeat()...drop()` Gremlin.
- Résolution d'un problème lié à l'étape `drop()` Gremlin.
- Résolution d'un problème lié aux filtres d'étiquette Gremlin.
- Résolution d'un problème lié aux délais d'expiration de requête Gremlin.

Mises à jour du moteur Amazon Neptune 02/07/2019

IMPORTANT : CETTE VERSION DU MOTEUR EST MAINTENANT OBSOLÈTE

À partir du 27 avril 2021, plus aucune instance ne sera créée avec cette version du moteur.

Défauts corrigés dans cette version du moteur

- Un bogue, qui empêchait l'optimisation de certains modèles dont le nom et la valeur de propriété étaient liés, a été corrigé.

Versions antérieures de Neptune Engine

Rubriques

- Mises à jour du moteur Amazon Neptune 12/06/2019
- Mises à jour du moteur Amazon Neptune 01/05/2019
- Mises à jour du moteur Amazon Neptune 21/01/2019
- Mises à jour du moteur Amazon Neptune 19/11/2018
- Mises à jour du moteur Amazon Neptune 08/11/2018
- Mises à jour du moteur Amazon Neptune 29/10/2018
- Mises à jour du moteur Amazon Neptune 06/09/2018
- Mises à jour du moteur Amazon Neptune 24/07/2018
- Mises à jour du moteur Amazon Neptune 22/06/2018

Mises à jour du moteur Amazon Neptune 12/06/2019

Version : 1.0.1.0.200310.0

Amazon Neptune 1.0.1.0.200310.0 est disponible globalement. Tous les nouveaux clusters de bases de données Neptune, y compris ceux restaurés à partir d'instantanés, seront créés dans 1.0.1.0.200310.0 une fois la mise à jour du moteur terminée pour cette région.

Les clusters existants peuvent être mis immédiatement à niveau vers cette version à l'aide des opérations de base de données sur la console ou à l'aide du kit SDK. Vous pouvez utiliser la commande CLI suivante pour mettre immédiatement à niveau un cluster de bases de données vers cette version :

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Les clusters de bases de données Neptune seront automatiquement mis à niveau vers la version 1.0.1.0.200310.0 du moteur pendant les fenêtres de maintenance du système. L'horaire d'application des mises à jour dépend de la région et du paramètre de fenêtre de maintenance configuré pour le cluster de bases de données, ainsi que du type de mise à jour.

> **Note**
>
> La fenêtre de maintenance de l'instance ne s'applique pas aux mises à jour du moteur.

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur toutes les instances figurant dans un cluster de bases de données, si bien que vous connaîtrez de 20-30 secondes à plusieurs minutes d'indisponibilité, après quoi vous pourrez reprendre l'utilisation de votre ou de vos clusters de bases de données. Vous pouvez consulter et modifier vos paramètres de fenêtre de maintenance dans la console Neptune.

Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Améliorations

- Un bogue, où l'insertion et la suppression simultanées d'une périphérie pouvait générer plusieurs périphéries portant le même ID, a été corrigé.
- Autres correctifs mineurs et améliorations.

Mises à jour du moteur Amazon Neptune 01/05/2019

Version : 1.0.1.0.200296.0

Amazon Neptune 1.0.1.0.200296.0 est disponible globalement. Tous les nouveaux clusters de bases de données Neptune, y compris ceux restaurés à partir d'instantanés, seront créés dans 1.0.1.0.200296.0 une fois la mise à jour du moteur terminée pour cette région.

Les clusters existants peuvent être mis immédiatement à niveau vers cette version à l'aide des opérations de base de données sur la console ou à l'aide du kit SDK. Vous pouvez utiliser la commande CLI suivante pour mettre immédiatement à niveau un cluster de bases de données vers cette version :

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Les clusters de bases de données Neptune seront automatiquement mis à niveau vers la version 1.0.1.0.200296.0 du moteur pendant les fenêtres de maintenance du système. L'horaire d'application des mises à jour dépend de la région et du paramètre de fenêtre de maintenance configuré pour le cluster de bases de données, ainsi que du type de mise à jour.

> **Note**
>
> La fenêtre de maintenance de l'instance ne s'applique pas aux mises à jour du moteur.

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur toutes les instances figurant dans un cluster de bases de données, si bien que vous connaîtrez de 20-30 secondes à plusieurs minutes d'indisponibilité, après quoi vous pourrez reprendre l'utilisation de votre ou de vos clusters de bases de données. Vous pouvez consulter et modifier vos paramètres de fenêtre de maintenance dans la console Neptune.

Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Améliorations

- Ajout de la nouvelle fonctionnalité `explain` aux requêtes SPARQL Neptune pour vous aider à visualiser le plan de requête et à prendre les mesures nécessaires pour l'optimiser si nécessaire. Pour plus d'informations, consultez SPARQL `explain`.
- Amélioration des performances et des rapports de SPARQL de diverses manières.
- Amélioration des performances et du comportement de Gremlin de diverses manières.
- Amélioration de l'expiration des requêtes `drop( )` de longue durée.
- Amélioration des performances des requêtes `otherV( )`.
- Ajout de deux champs aux informations renvoyées quand vous interrogez l'état Neptune d'un cluster ou d'une instance de base de données, notamment le numéro de version du moteur et l'heure de démarrage du cluster ou de l'instance. Consultez Statut d'une instance.

- L'API `Get-Status` du chargeur Neptune renvoie désormais un champ `startTime` qui enregistre l'heure de début d'une tâche de chargement.
- La commande de chargeur prend désormais un paramètre `parallelism` facultatif qui vous permet de limiter le nombre de threads que le chargeur utilise.

Mises à jour du moteur Amazon Neptune 21/01/2019

Version : 1.0.1.0.200267.0

Amazon Neptune 1.0.1.0.200267.0 est disponible globalement. Tous les nouveaux clusters de bases de données Neptune, y compris ceux restaurés à partir d'instantanés, seront créés dans 1.0.1.0.200267.0 une fois la mise à jour du moteur terminée pour cette région.

Les clusters existants peuvent être mis immédiatement à niveau vers cette version à l'aide des opérations de base de données sur la console ou à l'aide du kit SDK. Vous pouvez utiliser la commande CLI suivante pour mettre immédiatement à niveau un cluster de bases de données vers cette version :

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Les clusters de bases de données Neptune seront automatiquement mis à niveau vers la version 1.0.1.0.200267.0 du moteur pendant les fenêtres de maintenance du système. L'horaire d'application des mises à jour dépend de la région et du paramètre de fenêtre de maintenance configuré pour le cluster de bases de données, ainsi que du type de mise à jour.

> **Note**
>
> La fenêtre de maintenance de l'instance ne s'applique pas aux mises à jour du moteur.

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur toutes les instances figurant dans un cluster de bases de données, si bien que vous connaîtrez de 20-30 secondes à plusieurs minutes d'indisponibilité, après quoi vous pourrez reprendre l'utilisation de votre ou de vos clusters de bases de données. Vous pouvez consulter et modifier vos paramètres de fenêtre de maintenance dans la console Neptune.

Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Améliorations

- Neptune attend plus longtemps (dans le délai d'interrogation spécifié) que les conflits soient résolus. Cela réduit le nombre d'exceptions de modifications simultanées qui doivent être gérées par le client (voir Erreurs liées aux requêtes).
- Un problème qui amenait parfois l'application de la cardinalité par Gremlin à provoquer le redémarrage du moteur a été résolu.
- Amélioration des performances de Gremlin pour les requêtes répétées `emit.times`.
- Un problème de Gremlin par lequel `repeat.until` laissait passer des solutions `.emit` qui auraient dû être filtrées a été résolu.
- Amélioration de la gestion des erreurs dans Gremlin.

Mises à jour du moteur Amazon Neptune 19/11/2018

Version : 1.0.1.0.200264.0

Amazon Neptune 1.0.1.0.200264.0 est disponible globalement. Tous les nouveaux clusters de bases de données Neptune, y compris ceux restaurés à partir d'instantanés, seront créés dans 1.0.1.0.200264.0 une fois la mise à jour du moteur terminée pour cette région.

Les clusters existants peuvent être mis immédiatement à niveau vers cette version à l'aide des opérations de base de données sur la console ou à l'aide du kit SDK. Vous pouvez utiliser la commande CLI suivante pour mettre immédiatement à niveau un cluster de bases de données vers cette version :

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Les clusters de bases de données Neptune seront automatiquement mis à niveau vers la version 1.0.1.0.200264.0 du moteur pendant les fenêtres de maintenance du système. L'horaire d'application des mises à jour dépend de la région et du paramètre de fenêtre de maintenance configuré pour le cluster de bases de données, ainsi que du type de mise à jour.

> **Note**
>
> La fenêtre de maintenance de l'instance ne s'applique pas aux mises à jour du moteur.

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur toutes les instances figurant dans un cluster de bases de données, si bien que vous connaîtrez de 20-30 secondes à plusieurs minutes d'indisponibilité, après quoi vous pourrez reprendre l'utilisation de votre ou de vos clusters de bases de données. Vous pouvez consulter et modifier vos paramètres de fenêtre de maintenance dans la console Neptune.

Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Améliorations

- Ajout de la prise en charge de the section called “Indicateurs de requête”.
- Amélioration des messages d'erreur pour l'authentification IAM. Pour de plus amples informations, veuillez consulter the section called “Erreurs d'IAM”.
- Amélioration des performances des requêtes SPARQL avec un grand nombre de prédicats.
- Amélioration des performances de chemin de propriété SPARQL.
- Amélioration des performances de Gremlin pour les mutations conditionnelles, telles que le modèle `fold().coalesce(unfold(), …)` lorsqu'il est utilisé avec les étapes `addV()`, `addE()` et `property()`.
- Amélioration des performances de Gremlin pour les modulations `by()` et `sack()`.
- Amélioration des performances de Gremlin pour les étapes `group()` et `groupCount()`.
- Amélioration des performances de Gremlin pour les étapes `store()`, `sideEffect()` et `cap().unfold()`.
- Amélioration de la prise en charge pour les contraintes de propriétés de cardinalité unique Gremlin.
  - Amélioration de l'application de la cardinalité unique à des propriétés d'arc et des propriétés de sommet marquées comme des propriétés de cardinalité unique.
  - Introduction d'une erreur si des valeurs de propriétés supplémentaires sont spécifiées pour une propriété d'arc existante lors de tâches de chargement Neptune.

Mises à jour du moteur Amazon Neptune 08/11/2018

Version : 1.0.1.0.200258.0

Amazon Neptune 1.0.1.0.200258.0 est disponible globalement. Tous les nouveaux clusters de bases de données Neptune, y compris ceux restaurés à partir d'instantanés, seront créés dans 1.0.1.0.200258.0 une fois la mise à jour du moteur terminée pour cette région.

Les clusters existants peuvent être mis immédiatement à niveau vers cette version à l'aide des opérations de base de données sur la console ou à l'aide du kit SDK. Vous pouvez utiliser la commande CLI suivante pour mettre immédiatement à niveau un cluster de bases de données vers cette version :

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Les clusters de bases de données Neptune seront automatiquement mis à niveau vers la version 1.0.1.0.200258.0 du moteur pendant les fenêtres de maintenance du système. L'horaire d'application des mises à jour dépend de la région et du paramètre de fenêtre de maintenance configuré pour le cluster de bases de données, ainsi que du type de mise à jour.

> **Note**
>
> La fenêtre de maintenance de l'instance ne s'applique pas aux mises à jour du moteur.

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur toutes les instances figurant dans un cluster de bases de données, si bien que vous connaîtrez de 20-30 secondes à plusieurs minutes d'indisponibilité, après quoi vous pourrez reprendre l'utilisation de votre ou de vos clusters de bases de données. Vous pouvez consulter et modifier vos paramètres de fenêtre de maintenance dans la console Neptune.

Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Améliorations

- Ajout de la prise en charge de Indicateurs de requête SPARQL.
- Amélioration des performances des requêtes Exists SPARQL FILTER (NOT).
- Amélioration des performances des requêtes SPARQL DESCRIBE.
- Amélioration des performances pour le modèle de répétition jusqu'à dans Gremlin.
- Amélioration de la performance pour l'ajout d'arcs dans Gremlin.
- Résolution d'un problème où des requêtes SPARQL Update DELETE pouvaient échouer dans certains cas.
- Correction d'un problème de gestion des délais d'attente sur le serveur Gremlin WebSocket.

Mises à jour du moteur Amazon Neptune 29/10/2018

Version : 1.0.1.0.200255.0

Amazon Neptune 1.0.1.0.200255.0 est disponible globalement. Tous les nouveaux clusters de bases de données Neptune, y compris ceux restaurés à partir d'instantanés, seront créés dans 1.0.1.0.200255.0 une fois la mise à jour du moteur terminée pour cette région.

Les clusters existants peuvent être mis immédiatement à niveau vers cette version à l'aide des opérations de base de données sur la console ou à l'aide du kit SDK. Vous pouvez utiliser la commande CLI suivante pour mettre immédiatement à niveau un cluster de bases de données vers cette version :

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Les clusters de bases de données Neptune seront automatiquement mis à niveau vers la version 1.0.1.0.200255.0 du moteur pendant les fenêtres de maintenance du système. L'horaire d'application des mises à jour dépend de la région et du paramètre de fenêtre de maintenance configuré pour le cluster de bases de données, ainsi que du type de mise à jour.

> **Note**
>
> La fenêtre de maintenance de l'instance ne s'applique pas aux mises à jour du moteur.

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur toutes les instances figurant dans un cluster de bases de données, si bien que vous connaîtrez de 20-30 secondes à plusieurs minutes d'indisponibilité, après quoi vous pourrez reprendre l'utilisation de votre ou de vos clusters de bases de données. Vous pouvez consulter et modifier vos paramètres de fenêtre de maintenance dans la console Neptune.

Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Améliorations

- Ajout d'informations sur l'authentification IAM dans les journaux d'audit.
- Ajout de la prise en charge des informations d'identification temporaires à l'aide de rôles IAM et de profils d'instance.
- Ajout de la terminaison de WebSocket connexion pour l'authentification IAM lorsque l'autorisation est révoquée ou si l'utilisateur ou le rôle IAM est supprimé.
- Le nombre maximum de WebSocket connexions a été limité à 60 000 par instance.
- Amélioration des performances du chargement en bloc pour de plus petits types d'instance.
- Amélioration des performances pour les requêtes qui incluent les opérateurs `and()`, `or()`, `not()` et `drop()` dans Gremlin.
- L' NTriples analyseur rejette désormais les éléments non valides URIs, tels que ceux URIs contenant des espaces blancs.

Mises à jour du moteur Amazon Neptune 06/09/2018

Version : 1.0.1.0.200237.0

Amazon Neptune 1.0.1.0.200237.0 est disponible globalement. Tous les nouveaux clusters de bases de données Neptune, y compris ceux restaurés à partir d'instantanés, seront créés dans 1.0.1.0.200237.0 une fois la mise à jour du moteur terminée pour cette région.

Les clusters existants peuvent être mis immédiatement à niveau vers cette version à l'aide des opérations de base de données sur la console ou à l'aide du kit SDK. Vous pouvez utiliser la commande CLI suivante pour mettre immédiatement à niveau un cluster de bases de données vers cette version :

```
aws neptune apply-pending-maintenance-action \
```

```
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Les clusters de bases de données Neptune seront automatiquement mis à niveau vers la version 1.0.1.0.200237.0 du moteur pendant les fenêtres de maintenance du système. L'horaire d'application des mises à jour dépend de la région et du paramètre de fenêtre de maintenance configuré pour le cluster de bases de données, ainsi que du type de mise à jour.

> **Note**
>
> La fenêtre de maintenance de l'instance ne s'applique pas aux mises à jour du moteur.

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur toutes les instances figurant dans un cluster de bases de données, si bien que vous connaîtrez de 20-30 secondes à plusieurs minutes d'indisponibilité, après quoi vous pourrez reprendre l'utilisation de votre ou de vos clusters de bases de données. Vous pouvez consulter et modifier vos paramètres de fenêtre de maintenance dans la console Neptune.

Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Améliorations

- Résolution d'un problème où des requêtes `SPARQL COUNT(DISTINCT)` échouaient.
- Résolution d'un problème où les requêtes `COUNT`, `SUM` et `MIN` avec une clause `DISTINCT` manquaient de mémoire.
- Résolution d'un problème où le type de données `BLOB` entraînait l'échec d'une tâche de chargeur Neptune.
- Résolution d'un problème où les insertions en double entraînaient des échecs de transaction.
- Résolution d'un problème où les requêtes `DROP ALL` ne pouvaient pas être annulées.
- Résolution d'un problème où les clients Gremlin se bloquaient de façon intermittente.
- Mise à jour de tous les codes d'erreur pour les charges utiles plus volumineuses que 150 M à la valeur `HTTP 400`.
- Amélioration des performances et de la précision des single-triple-pattern `COUNT()` requêtes.

- Amélioration des performances des requêtes `SPARQL UNION` avec les clauses `BIND`.

Mises à jour du moteur Amazon Neptune 24/07/2018

Version : 1.0.1.0.200236.0

Amazon Neptune 1.0.1.0.200236.0 est disponible globalement. Tous les nouveaux clusters de bases de données Neptune, y compris ceux restaurés à partir d'instantanés, seront créés dans 1.0.1.0.200236.0 une fois la mise à jour du moteur terminée pour cette région.

Les clusters existants peuvent être mis immédiatement à niveau vers cette version à l'aide des opérations de base de données sur la console ou à l'aide du kit SDK. Vous pouvez utiliser la commande CLI suivante pour mettre immédiatement à niveau un cluster de bases de données vers cette version :

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Les clusters de bases de données Neptune seront automatiquement mis à niveau vers la version 1.0.1.0.200236.0 du moteur pendant les fenêtres de maintenance du système. L'horaire d'application des mises à jour dépend de la région et du paramètre de fenêtre de maintenance configuré pour le cluster de bases de données, ainsi que du type de mise à jour.

> **Note**
>
> La fenêtre de maintenance de l'instance ne s'applique pas aux mises à jour du moteur.

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur toutes les instances figurant dans un cluster de bases de données, si bien que vous connaîtrez de 20-30 secondes à plusieurs minutes d'indisponibilité, après quoi vous pourrez reprendre l'utilisation de votre ou de vos clusters de bases de données. Vous pouvez consulter et modifier vos paramètres de fenêtre de maintenance dans la console Neptune.

Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Améliorations

- Mise à jour de la sérialisation SPARQL pour le type de données `xsd:string`. xsd:string n'est plus inclus dans la sérialisation JSON, qui est désormais cohérente avec d'autres formats de sortie.
- Correction du traitement de l'infini `xsd:double`/`xsd:float`. Les valeurs `-INF`, `NaN`, et `INF` sont désormais reconnues et traitées correctement dans tous les formats de chargeur de données SPARQL, SPARQL 1.1 UPDATE et SPARQL 1.1 Query.
- Résolution d'un problème en raison duquel une requête Gremlin avec des valeurs de chaîne vides échoue de manière inattendue.
- Résolution d'un problème en raison duquel `aggregate()` et `cap()` sur un graphique vide échoue de manière inattendue.
- Résolution d'un problème en raison duquel des réponses d'erreur incorrectes sont renvoyées pour Gremlin lorsque la spécification de cardinalité n'est pas valide, par exemple `.property(set,id,'10')` et `.property(single,id,'10')`.
- Correction d'un problème en raison duquel une syntaxe Gremlin non valide était renvoyée sous forme de InternalFailureException.
- Correction de l'orthographe dans `TimeLimitExceeededException` en `TimeLimitExceededException`, dans des messages d'erreur.
- Modification de la réponse des points de terminaison SPARQL et GREMLIN en une réponse cohérente lorsqu'aucun script n'est fourni.
- Clarification des messages d'erreur pour un trop grand nombre de demandes simultanées.

Mises à jour du moteur Amazon Neptune 22/06/2018

Version : 1.0.1.0.200233.0

Amazon Neptune 1.0.1.0.200233.0 est disponible globalement. Tous les nouveaux clusters de bases de données Neptune, y compris ceux restaurés à partir d'instantanés, seront créés dans 1.0.1.0.200233.0 une fois la mise à jour du moteur terminée pour cette région.

Les clusters existants peuvent être mis immédiatement à niveau vers cette version à l'aide des opérations de base de données sur la console ou à l'aide du kit SDK. Vous pouvez utiliser la commande CLI suivante pour mettre immédiatement à niveau un cluster de bases de données vers cette version :

```
aws neptune apply-pending-maintenance-action \
```

```
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Les clusters de bases de données Neptune seront automatiquement mis à niveau vers la version 1.0.1.0.200233.0 du moteur pendant les fenêtres de maintenance du système. L'horaire d'application des mises à jour dépend de la région et du paramètre de fenêtre de maintenance configuré pour le cluster de bases de données, ainsi que du type de mise à jour.

> **Note**
>
> La fenêtre de maintenance de l'instance ne s'applique pas aux mises à jour du moteur.

Les mises à jour sont appliquées simultanément à toutes les instances figurant dans un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur toutes les instances figurant dans un cluster de bases de données, si bien que vous connaîtrez de 20-30 secondes à plusieurs minutes d'indisponibilité, après quoi vous pourrez reprendre l'utilisation de votre ou de vos clusters de bases de données. Vous pouvez consulter et modifier vos paramètres de fenêtre de maintenance dans la console Neptune.

Si vous avez des questions ou des préoccupations, l'équipe de AWS support est disponible sur les forums communautaires et via le support AWS Premium.

Améliorations

- Résolution d'un problème où l'émission en succession rapide d'un grand nombre de demandes de chargement en masse provoquait une erreur.
- Correction d'un problème dépendant des données à cause duquel une requête pouvait échouer avec un InternalServerError. L'exemple suivant montre le type de requête concerné.

```
g.V("my-id123").as("start").outE("knows").has("edgePropertyKey1",
 P.gt(0)).as("myedge").inV()
              .as("end").select("start", "end", "myedge").by("vertexPropertyKey1")
              .by("vertexPropertyKey1").by("edgePropertyKey1")
```

- Correction d'un problème en raison duquel un client Java Gremlin ne pouvait pas se connecter au serveur en utilisant la même WebSocket connexion après l'expiration du délai d'une requête de longue durée.

- Correction d'un problème en raison duquel les séquences échappées contenues dans le cadre de la requête Gremlin via HTTP ou des requêtes basées sur des chaînes via la WebSocket connexion n'étaient pas traitées correctement.

Présentation de l'utilisation d'Amazon Neptune APIs

La direction d'Amazon Neptune APIs fournit un support SDK pour créer, gérer et supprimer des clusters et des instances de base de données Neptune, tandis que les données Neptune APIs fournissent un support SDK pour charger des données dans votre graphe, exécuter des requêtes, obtenir des informations sur les données de votre graphe et exécuter des opérations d'apprentissage automatique. Ils APIs sont disponibles dans toutes les langues du SDK. Elles signent automatiquement les demandes d'API et simplifient donc considérablement l'intégration de Neptune dans les applications.

Cette page fournit des informations sur leur utilisation APIs.

Actions IAM dont les noms diffèrent de leurs équivalents dans les kits SDK d'API de données Neptune

Lorsque vous appelez une méthode d'API Neptune sur un cluster sur lequel l'authentification IAM est activée, une politique IAM doit être associée à l'utilisateur ou au rôle effectuant les appels. Celle-ci fournit les autorisations nécessaires pour les actions que vous souhaitez effectuer. Vous définissez ces autorisations dans la politique à l'aide des actions IAM correspondantes. Vous pouvez également restreindre les actions pouvant être effectuées à l'aide des clés de condition IAM.

La plupart des actions IAM portent le même nom que les méthodes d'API auxquelles elles correspondent, mais certaines méthodes de l'API de données portent des noms différents, car certaines sont partagées par plusieurs méthodes. Le tableau ci-dessous répertorie les méthodes de l'API de données et les actions IAM correspondantes :

Nom de l'opération d'API de données	Correspondances IAM	
CancelGremlinQuery(annuler_gremlin_query)	Action : `neptune-db: CancelQuery`	
CancelLoaderJob(annuler_loader_job)	Action : `neptune-db:CancelLoaderJob`	

Nom de l'opération d'API de données	Correspondances IAM	
Annuler MLData ProcessingJob (cancel_ml_data_processing_job)	Action : `neptune-db:CancelMLDataProcessingJob`	
Annuler MLModel TrainingJob (cancel_ml_model_training_job)	Action : `neptune-db:CancelMLModelTrainingJob`	
CancelOpenCypherQuery(annuler_open_cypher_query)	Action : `neptune-db: **CancelQuery**`	
Créer MLEndpoint (create_ml_endpoint)	Action : `neptune-db:CreateMLEndpoint`	
Supprimer MLEndpoint (delete_ml_endpoint)	Action : `neptune-db:DeleteMLEndpoint`	
DeletePropertygraphStatistics(supprimer_propertygraph_statistics)	Action : `neptune-db: **DeleteStatistics**`	
DeleteSparqlStatistics(supprimer_sparql_statistics)	Action : `neptune-db: **DeleteStatistics**`	
ExecuteFastResetexecute_fast_reset ()	Action : `neptune-db: **ResetDatabase**`	

Nom de l'opération d'API de données	Correspondances IAM	
ExecuteGremlinExplainQuery(exécute_gremlin_explain_query)	Actions : • neptune-db: **ReadDataViaQuery** • neptune-db: **WriteDataViaQuery** • neptune-db: **DeleteDataViaQuery** Clé de condition : neptune-db:QueryLanguage:Gremlin	
ExecuteGremlinProfileQuery(exécute_gremlin_profile_query)	Action : neptune-db: **ReadDataViaQuery** Clé de condition : neptune-db:QueryLanguage:Gremlin	
ExecuteGremlinQuery(exécute_gremlin_query)	Actions : • neptune-db: **ReadDataViaQuery** • neptune-db: **WriteDataViaQuery** • neptune-db: **DeleteDataViaQuery** Clé de condition : neptune-db:QueryLanguage:Gremlin	

Nom de l'opération d'API de données	Correspondances IAM	
ExecuteOpenCypherExplainQuery(exécute_open_cypher_explain_query)	Action : `neptune-db: ReadDataViaQuery` Clé de condition : `neptune-db:QueryLanguage:OpenCypher`	
ExecuteOpenCypherQuery(exécute_open_cypher_query)	Actions : • `neptune-db: ReadDataViaQuery` • `neptune-db: WriteDataViaQuery` • `neptune-db: DeleteDataViaQuery` Clé de condition : `neptune-db:QueryLanguage:OpenCypher`	
GetEngineStatus(get_engine_status)	Action : `neptune-db:GetEngineStatus`	
GetGremlinQueryStatus(get_gremlin_query_status)	Action : `neptune-db: :GetQueryStatus` Clé de condition : `neptune-db:QueryLanguage:Gremlin`	
GetLoaderJobStatus(get_loader_job_status)	Action : `neptune-db:GetLoaderJobStatus`	

Nom de l'opération d'API de données	Correspondances IAM	
Obtenir MLData ProcessingJob (get_ml_data_processing_job)	Action : `neptune-db:` **`GetMLDataProcessingJobStatus`**	
Obtenir MLEndpoint (get_ml_endpoint)	Action : `neptune-db:` **`GetMLEndpointStatus`**	
Obtenir MLModel TrainingJob (get_ml_model_training_job)	Action : `neptune-db:` **`GetMLModelTrainingJobStatus`**	
Obtenir MLModel TransformJob (get_ml_model_transform_job)	Action : `neptune-db:` **`GetMLModelTransformJobStatus`**	
GetOpenCypherQueryStatus(get_open_cypher_query_status)	Action : `neptune-db:` **`:GetQueryStatus`** Clé de condition : `neptune-db:QueryLanguage:OpenCypher`	
GetPropertygraphStatistics(get_propertygraph_statistics)	Action : `neptune-db:` **`GetStatisticsStatus`**	
GetPropertygraphStream(get_propertygraph_stream)	Action : `neptune-db:` **`GetStreamRecords`** Clés de condition : • `neptune-db:QueryLanguage:Gremlin` • `neptune-db:QueryLanguage:OpenCypher`	

Nom de l'opération d'API de données	Correspondances IAM	
GetPropertygraphSummary(get_propertygraph_summary)	Action : `neptune-db:` **`GetGraphSummary`**	
Obtenir le RDFGraph résumé (get_rdf_graph_summary)	Action : `neptune-db:` **`GetGraphSummary`**	
GetSparqlStatistics(get_sparql_statistics)	Action : `neptune-db:` **`GetStatisticsStatus`**	
GetSparqlStream(get_sparql_stream)	Action : `neptune-db:` **`:GetStreamRecords`** Clé de condition : `neptune-db:QueryLanguage:Sparql`	
ListGremlinQueries(liste_gremlin_requêtes)	Action : `neptune-db:` **`:GetQueryStatus`** Clé de condition : `neptune-db:QueryLanguage:Gremlin`	
Liste MLEndpoints (list_ml_endpoints)	Action : `neptune-db:ListMLEndpoints`	
Liste MLModel TrainingJobs (list_ml_model_training_jobs)	Action : `neptune-db:ListMLModelTrainingJobs`	
Liste MLModel TransformJobs (list_ml_model_transform_jobs)	Action : `neptune-db:ListMLModelTransformJobs`	

Nom de l'opération d'API de données	Correspondances IAM	
ListOpenCypherQueries(list_open_cypher_queries)	Action : `neptune-db: :GetQueryStatus` Clé de condition : `neptune-db:QueryLanguage:OpenCypher`	
ManagePropertygraphStatistics(manage_propertygraph_statistics)	Action : `neptune-db: ManageStatistics`	
ManageSparqlStatistics(manage_sparql_statistics)	Action : `neptune-db: ManageStatistics`	
StartLoaderJob(start_loader_job)	Action : `neptune-db:StartLoaderJob`	
Démarrer MLModel DataProcessingJob (start_ml_data_processing_job)	Action : `neptune-db:StartMLModelDataProcessingJob`	
Démarrer MLModel TrainingJob (start_ml_model_training_job)	Action : `neptune-db:StartMLModelTrainingJob`	
Démarrer MLModel TransformJob (start_ml_model_transform_job)	Action : `neptune-db:StartMLModelTransformJob`	

Référence de l'API de gestion Amazon Neptune

Ce chapitre décrit les opérations d'API Neptune que vous pouvez utiliser pour gérer le cluster de bases de données Neptune.

Neptune opère sur des clusters de serveurs de base de données qui sont connectés dans une topologie de réplication. Par conséquent, la gestion de Neptune implique souvent de déployer des modifications sur plusieurs serveurs et de s'assurer que tous les réplicas Neptune suivent le rythme du serveur principal.

Étant donné que Neptune met à l'échelle le stockage sous-jacent en toute transparence à mesure que vos données augmentent, la gestion de Neptune exige relativement peu de gestion au niveau du stockage sur disque. De la même manière, étant donné que Neptune effectue automatiquement des sauvegardes continues, un cluster Neptune nécessite peu de planification ou de temps d'arrêt pour la réalisation des sauvegardes.

Table des matières

- API pour clusters de bases de données Neptune
  - Créer DBCluster (action)
  - Supprimer DBCluster (action)
  - Modifier DBCluster (action)
  - Démarrer DBCluster (action)
  - Arrêter DBCluster (action)
  - AddRoleToDBCluster (action)
  - RemoveRoleFromDBCluster (action)
  - Basculement DBCluster (action)
  - PromoteReadReplicaDBCluster (action)
  - Décrire DBClusters (action)
  - Structures :
  - DBCluster (structure)
  - DBClusterMembre (structure)
  - DBClusterRôle (structure)
  - CloudwatchLogsExportConfiguration (structure)
  - PendingCloudwatchLogsExports (structure)

- ClusterPendingModifiedValues (structure)
- API de base de données globale Neptune
  - CreateGlobalCluster (action)
  - DeleteGlobalCluster (action)
  - ModifyGlobalCluster (action)
  - DescribeGlobalClusters (action)
  - FailoverGlobalCluster (action)
  - RemoveFromGlobalCluster (action)
  - Structures :
  - GlobalCluster (structure)
  - GlobalClusterMember (structure)
- API d'instances Neptune
  - Créer DBInstance (action)
  - Supprimer DBInstance (action)
  - Modifier DBInstance (action)
  - Redémarrer DBInstance (action)
  - Décrire DBInstances (action)
  - DescribeOrderableDBInstanceOptions (action)
  - DescribeValidDBInstanceModifications (action)
  - Structures :
  - DBInstance (structure)
  - DBInstanceStatusInfo (structure)
  - DBInstanceOption commandable (structure)
  - PendingModifiedValues (structure)
  - ValidStorageOptions (structure)
  - Valide DBInstance ModificationsMessage (structure)
- API de paramètres Neptune
  - Copier DBParameter le groupe (action)
  - Copier DBCluster ParameterGroup (action)
  - Créer un DBParameter groupe (action)

- Créer DBCluster ParameterGroup (action)
- Supprimer le DBParameter groupe (action)
- Supprimer DBCluster ParameterGroup (action)
- Modifier DBParameter le groupe (action)
- Modifier DBCluster ParameterGroup (action)
- Réinitialiser DBParameter le groupe (action)
- Réinitialiser DBCluster ParameterGroup (action)
- Décrire DBParameters (action)
- Décrire DBParameter les groupes (action)
- Décrire DBCluster les paramètres (action)
- Décrire DBCluster ParameterGroups (action)
- DescribeEngineDefaultParameters (action)
- DescribeEngineDefaultClusterParameters (action)
- Structures :
- Parameter (structure)
- DBParameterGroupe (structure)
- DBClusterParameterGroup (structure)
- DBParameterGroupStatus (structure)

- API de sous-réseau Neptune
  - Créer un DBSubnet groupe (action)
  - Supprimer le DBSubnet groupe (action)
  - Modifier DBSubnet le groupe (action)
  - Décrire DBSubnet les groupes (action)
  - Structures :
  - Subnet (structure)
  - DBSubnetGroupe (structure)
- API d'instantanés Neptune
  - Créer DBCluster un instantané (action)
  - Supprimer DBCluster un instantané (action)
  - Copier DBCluster un instantané (action)

- Modifier DBCluster SnapshotAttribute (action)
- Restaurer DBCluster FromSnapshot (action)
- Restaurer DBCluster ToPointInTime (action)
- Décrire les DBCluster instantanés (action)
- Décrire DBCluster SnapshotAttributes (action)
- Structures :
- DBClusterInstantané (structure)
- DBClusterSnapshotAttribute (structure)
- DBClusterSnapshotAttributesResult (structure)

- API d'événements Neptune
  - CreateEventSubscription (action)
  - DeleteEventSubscription (action)
  - ModifyEventSubscription (action)
  - DescribeEventSubscriptions (action)
  - AddSourceIdentifierToSubscription (action)
  - RemoveSourceIdentifierFromSubscription (action)
  - DescribeEvents (action)
  - DescribeEventCategories (action)
  - Structures :
  - Event (structure)
  - EventCategoriesMap (structure)
  - EventSubscription (structure)
- Autre Neptune APIs
  - AddTagsToResource (action)
  - ListTagsForResource (action)
  - RemoveTagsFromResource (action)
  - ApplyPendingMaintenanceAction (action)
  - DescribePendingMaintenanceActions (action)
  - Décrire DBEngine les versions (action)
  - Structures :

- DBEngineVersion (structure)
- EngineDefaults (structure)
- PendingMaintenanceAction (structure)
- ResourcePendingMaintenanceActions (structure)
- UpgradeTarget (structure)
- Tag (structure)

- Types de données Neptune courants
  - AvailabilityZone (structure)
  - DBSecurityGroupMembership (structure)
  - DomainMembership (structure)
  - DoubleRange (structure)
  - Endpoint (structure)
  - Filter (structure)
  - Range (structure)
  - ServerlessV2 ScalingConfiguration (structure)
  - ServerlessV2 ScalingConfigurationInfo (structure)
  - Timezone (structure)
  - VpcSecurityGroupMembership (structure)
- Neptune : exceptions spécifiques à un individu APIs
  - AuthorizationAlreadyExistsFault (structure)
  - AuthorizationNotFoundFault (structure)
  - AuthorizationQuotaExceededFault (structure)
  - CertificateNotFoundFault (structure)
  - DBClusterAlreadyExistsFault (structure)
  - DBClusterNotFoundFault (structure)
  - DBClusterParameterGroupNotFoundFault (structure)
  - DBClusterQuotaExceededFault (structure)
  - DBClusterRoleAlreadyExistsFault (structure)
  - DBClusterRoleNotFoundFault (structure)
  - DBClusterRoleQuotaExceededFault (structure)

- DBClusterSnapshotAlreadyExistsFault (structure)
- DBClusterSnapshotNotFoundFault (structure)
- DBInstanceAlreadyExistsFault (structure)
- DBInstanceNotFoundFault (structure)
- DBLogFileNotFoundFault (structure)
- DBParameterGroupAlreadyExistsFault (structure)
- DBParameterGroupNotFoundFault (structure)
- DBParameterGroupQuotaExceededFault (structure)
- DBSecurityGroupAlreadyExistsFault (structure)
- DBSecurityGroupNotFoundFault (structure)
- DBSecurityGroupNotSupportedFault (structure)
- DBSecurityGroupQuotaExceededFault (structure)
- DBSnapshotAlreadyExistsFault (structure)
- DBSnapshotNotFoundFault (structure)
- DBSubnetGroupAlreadyExistsFault (structure)
- DBSubnetGroupDoesNotCoverEnoughAZs (structure)
- DBSubnetGroupNotAllowedFault (structure)
- DBSubnetGroupNotFoundFault (structure)
- DBSubnetGroupQuotaExceededFault (structure)
- DBSubnetQuotaExceededFault (structure)
- DBUpgradeDependencyFailureFault (structure)
- DomainNotFoundFault (structure)
- EventSubscriptionQuotaExceededFault (structure)
- GlobalClusterAlreadyExistsFault (structure)
- GlobalClusterNotFoundFault (structure)
- GlobalClusterQuotaExceededFault (structure)
- InstanceQuotaExceededFault (structure)
- Insuffisant DBCluster CapacityFault (structure)
- Insuffisant DBInstance CapacityFault (structure)
- InsufficientStorageClusterCapacityFault (structure)

- Non valide DBCluster EndpointStateFault (structure)
- Non valide DBCluster SnapshotStateFault (structure)
- Non valide DBCluster StateFault (structure)
- Non valide DBInstance StateFault (structure)
- Non valide DBParameter GroupStateFault (structure)
- Non valide DBSecurity GroupStateFault (structure)
- Non valide DBSnapshot StateFault (structure)
- Non valide DBSubnet GroupFault (structure)
- Non valide DBSubnet GroupStateFault (structure)
- Non valide DBSubnet StateFault (structure)
- InvalidEventSubscriptionStateFault (structure)
- InvalidGlobalClusterStateFault (structure)
- InvalidOptionGroupStateFault (structure)
- InvalidRestoreFault (structure)
- InvalidSubnet (structure)
- Non valide VPCNetwork StateFault (structure)
- KMSKeyNotAccessibleFault (structure)
- OptionGroupNotFoundFault (structure)
- PointInTimeRestoreNotEnabledFault (structure)
- ProvisionedIopsNotAvailableInAZFault (structure)
- ResourceNotFoundFault (structure)
- SNSInvalidTopicFault (structure)
- SNSNoAuthorizationFault (structure)
- SNSTopicArnNotFoundFault (structure)
- SharedSnapshotQuotaExceededFault (structure)
- SnapshotQuotaExceededFault (structure)
- SourceNotFoundFault (structure)
- StorageQuotaExceededFault (structure)
- StorageTypeNotSupportedFault (structure)
- SubnetAlreadyInUse (structure)

- SubscriptionAlreadyExistFault (structure)
- SubscriptionCategoryNotFoundFault (structure)
- SubscriptionNotFoundFault (structure)

API pour clusters de bases de données Neptune

Actions :

- Créer DBCluster (action)
- Supprimer DBCluster (action)
- Modifier DBCluster (action)
- Démarrer DBCluster (action)
- Arrêter DBCluster (action)
- AddRoleToDBCluster (action)
- RemoveRoleFromDBCluster (action)
- Basculement DBCluster (action)
- PromoteReadReplicaDBCluster (action)
- Décrire DBClusters (action)

Structures :

- DBCluster (structure)
- DBClusterMembre (structure)
- DBClusterRôle (structure)
- CloudwatchLogsExportConfiguration (structure)
- PendingCloudwatchLogsExports (structure)
- ClusterPendingModifiedValues (structure)

Créer DBCluster (action)

> Le nom de la AWS CLI pour cette API est :`create-db-cluster`.

Crée un cluster de base de données Amazon Neptune.

Vous pouvez utiliser le paramètre `ReplicationSourceIdentifier` pour créer le cluster de base de données en tant que réplica en lecture d'un autre cluster de base de données ou d'une instance de base de données Amazon Neptune.

Notez que lorsque vous créez un nouveau cluster en utilisant `CreateDBCluster` directement, la protection contre la suppression est désactivée par défaut (lorsque vous créez un nouveau cluster de production dans la console, la protection contre la suppression est activée par défaut). Vous ne pouvez supprimer un cluster de base de données que si son champ `DeletionProtection` est défini sur `false`.

Demande

- AvailabilityZones (dans la CLI : `--availability-zones`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des zones de EC2 disponibilité dans lesquelles les instances du cluster de base de données peuvent être créées.
- BackupRetentionPeriod(dans la CLI :`--backup-retention-period`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre de jours de conservation des sauvegardes automatiques. Vous devez spécifier une valeur minimale de 1.

  Par défaut : 1

  Contraintes :

  - Doit être une valeur comprise entre 1 et 35
- CopyTagsToSnapshot(dans la CLI :`--copy-tags-to-snapshot`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur *`true`*, les balises sont copiées dans n'importe quel instantané du cluster de bases de données créé.
- DatabaseName (dans la CLI : `--database-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de votre base de données comprenant au maximum 64 caractères alphanumériques. Si vous ne fournissez pas de nom, Amazon Neptune ne crée pas de base de données dans le cluster de base de données que vous créez.

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du cluster de bases de données. Ce paramètre est stocké sous la forme d'une chaîne en lettres minuscules.

  Contraintes :

  - Doit contenir entre 1 et 63 lettres, chiffres ou traits d'union.
  - Le premier caractère doit être une lettre.
  - Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs.

  Exemple : `my-cluster1`
- DBClusterParameterGroupName (dans la CLI : `--db-cluster-parameter-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de cluster DB à associer à ce cluster DB. Si cet argument est omis, la valeur par défaut est utilisée.

  Contraintes :

  - S'il est fourni, il doit correspondre au nom d'un existant DBClusterParameterGroup.
- DBSubnetGroupName (dans la CLI : `--db-subnet-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Groupe de sous-réseaux de base de données à associer à ce cluster de base de données.

  Contraintes : Doit correspondre au nom d'un DBSubnet groupe existant. Impossible de conserver le nom par défaut.

  Exemple : `mySubnetgroup`
- DeletionProtection(dans la CLI :`--deletion-protection`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Valeur qui indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée. Par défaut, la protection contre la suppression est désactivée.
- EnableCloudwatchLogsExports (dans la CLI : `--enable-cloudwatch-logs-exports`) : chaîne de type : `string` (chaîne encodée en UTF-8).

Liste des types de journaux que ce cluster de base de données doit exporter vers CloudWatch Logs. Les types de journaux valides sont : `audit` (pour publier les journaux d'audit) et `slowquery` (pour publier les journaux de requêtes lentes). Consultez la section Publication des journaux Neptune sur Amazon CloudWatch Logs.

- EnableIAMDatabaseAuthentication(dans la CLI :`--enable-iam-database-authentication`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur `true`, active l'authentification Amazon Identity and Access Management (IAM) pour l'ensemble du cluster de bases de données (ce qui ne peut pas être défini au niveau de l'instance).

  Par défaut: `false`.

- Engine (dans la CLI : `--engine`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du moteur de base de données à utiliser pour ce cluster de base de données.

  Valeurs valides : `neptune`

- EngineVersion (dans la CLI : `--engine-version`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Numéro de version du moteur de base de données à utiliser pour le nouveau cluster de bases de données.

  Exemple : `1.2.1.0`

- GlobalClusterIdentifier(dans la CLI :`--global-cluster-identifier`) — a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  ID de la base de données globale Neptune dans laquelle ce nouveau cluster de bases de données doit être ajouté.

- KmsKeyId (dans la CLI : `--kms-key-id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de clé Amazon KMS pour un cluster de bases de données chiffré.

  L'identifiant de clé KMS est l'Amazon Resource Name (ARN) de la clé de chiffrement KMS. Si vous créez un cluster de bases de données avec le compte Amazon qui possède la clé de chiffrement

KMS utilisée pour chiffrer le nouveau cluster de bases de données, vous pouvez utiliser l'alias de clé KMS au lieu de l'ARN de la clé de chiffrement KMS.

Si aucune clé de chiffrement n'est spécifiée dans `KmsKeyId` :

- Si `ReplicationSourceIdentifier` identifie une source chiffrée, Amazon Neptune utilise la clé de chiffrement qui a servi à chiffrer la source. Sinon, Amazon Neptune utilise votre clé de chiffrement par défaut.
- Si le paramètre `StorageEncrypted` a la valeur true et que `ReplicationSourceIdentifier` n'est pas spécifié, Amazon Neptune utilise votre clé de chiffrement par défaut.

Amazon KMS crée la clé de chiffrement par défaut pour votre compte Amazon. Votre compte Amazon a une clé de chiffrement par défaut différente pour chaque région Amazon.

Si vous créez un réplica en lecture d'un cluster de bases de données chiffré dans une autre région Amazon, vous devez définir `KmsKeyId` sur un ID de clé KMS qui est valide dans la région Amazon de destination. Cette clé est utilisée pour chiffrer le réplica en lecture dans cette région Amazon.

- Port(dans la CLI :`--port`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Numéro de port au niveau duquel les instances du cluster de base de données acceptent les connexions.

  Par défaut : `8182`

- PreferredBackupWindow (dans la CLI : `--preferred-backup-window`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Plage de temps quotidienne au cours de laquelle les sauvegardes automatiques sont créées si cette fonctionnalité est activée via le paramètre `BackupRetentionPeriod`.

  Par défaut, une fenêtre de 30 minutes est sélectionnée de manière aléatoire sur un bloc horaire de 8 heures pour chaque région Amazon. Pour voir les blocs horaires disponibles, consultez Fenêtre de maintenance de Neptune dans le Guide de l'utilisateur Amazon Neptune.

  Contraintes :

  - Doit être au format `hh24:mi-hh24:mi`.
  - Doit être exprimée en heure UTC (Universal Coordinated Time).
  - Ne doit pas être en conflit avec la fenêtre de maintenance préférée.

  - Doit être de 30 minutes minimum.
- PreferredMaintenanceWindow (dans la CLI : `--preferred-maintenance-window`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.

  Format : `ddd:hh24:mi-ddd:hh24:mi`

  Par défaut, une fenêtre de 30 minutes est sélectionnée de manière aléatoire dans un bloc de 8 heures pour chaque région Amazon, se produisant un jour choisi au hasard dans la semaine. Pour voir les blocs horaires disponibles, consultez Fenêtre de maintenance de Neptune dans le Guide de l'utilisateur Amazon Neptune.

  Jours valides : Mon, Tue, Wed, Thu, Fri, Sat, Sun.

  Contraintes : fenêtre minimale de 30 minutes.
- PreSignedUrl (dans la CLI : `--pre-signed-url`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Ce paramètre n'est actuellement pas pris en charge.
- ReplicationSourceIdentifier (dans la CLI : `--replication-source-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'instance de base de données ou du cluster de base de données source, si ce cluster de base de données est créé en tant que réplica en lecture.
- ServerlessV2ScalingConfiguration (dans la CLI : `--serverless-v2-scaling-configuration`) : objet Serverless V2 ScalingConfiguration.

  Contient la configuration de mise à l'échelle d'un cluster de bases de données Neptune sans serveur.

  Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.
- StorageEncrypted(dans la CLI :`--storage-encrypted`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si le cluster de base de données est chiffré.

- StorageType (dans la CLI : `--storage-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de stockage pour le cluster de bases de données.

  Valeurs valides :

  - **`standard`**— (valeur par défaut) Configure un stockage de base de données rentable pour les applications dont l' I/O utilisation est modérée à faible. Lorsqu'il est défini sur `standard`, le type de stockage n'est pas renvoyé dans la réponse.
  - **`iopt1`**— Permet un stockage optimisé pour les E/S, conçu pour répondre aux besoins de I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O débit.

    Le stockage optimisé pour les E/S Neptune n'est disponible qu'à partir de la version 1.3.0.0 du moteur.
- Tags (dans la CLI : `--tags`) : tableau d'objets Tag.

  Balises à affecter au nouveau cluster de base de données.
- VpcSecurityGroupIds (dans la CLI : `--vpc-security-group-ids`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des groupes de sécurité EC2 VPC à associer à ce cluster de base de données.

Réponse

Contient les détails d'un cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans the section called "Décrivez DBClusters".

- AllocatedStorage— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  `AllocatedStorage` renvoie toujours la valeur 1, car la taille de stockage d'un cluster de bases de données Neptune n'est pas fixe ; elle s'ajuste automatiquement en fonction des besoins.
- AssociatedRoles : tableau d'objets DBClusterRôle.

  Fournit la liste des rôles Amazon Identity and Access Management (IAM) associés au cluster de bases de données. Les rôles IAM associés à un cluster de bases de données autorisent celui-ci à accéder aux autres services Amazon en votre nom.

- AutomaticRestartTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Heure à laquelle le cluster de bases de données sera automatiquement redémarré.
- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du cluster de base de données peuvent être créées.
- BacktrackConsumedChangeRecords— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.
- BacktrackWindow— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.
- BackupRetentionPeriod— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.
- Capacity— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Non pris en charge par Neptune.
- CloneGroupId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifie le groupe de clones auquel est associé le cluster de bases de données.
- ClusterCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).
- CopyTagsToSnapshot— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur *`true`*, les balises sont copiées dans n'importe quel instantané du cluster de bases de données créé.
- CrossAccountClone— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur `true`, le cluster de bases de données peut être cloné sur plusieurs comptes.

- DatabaseName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nom de la base de données initiale de ce cluster de bases de données qui a été fourni au moment de la création, dans la mesure où un nom a été spécifié au moment de créer le cluster de bases de données. Ce même nom est renvoyé pendant la durée de vie du cluster de bases de données.

- DBClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du cluster de bases de données.

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de cluster de bases de données fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie un cluster de bases de données.

- DBClusterMembers : tableau d'objets DBClusterMembre.

  Fournit la liste des instances qui composent le cluster de bases de données.

- DBClusterParameterGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du groupe de paramètres de cluster de base de données pour le cluster de base de données.

- DbClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à la région Amazon du cluster de bases de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS du cluster de base de données est consultée.

- DBSubnetGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie les informations sur le groupe de sous-réseaux associé au cluster de bases de données, notamment le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée.

- EarliestBacktrackTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

Non pris en charge par Neptune.

- EarliestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure la plus proche à laquelle une base de données peut être restaurée par point-in-time restauration.

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux que ce cluster de base de données est configuré pour exporter vers CloudWatch Logs. Les types de journaux valides sont les suivants : `audit` (pour publier les journaux d'audit CloudWatch) et slowquery (pour publier les journaux de requêtes lentes). CloudWatch Consultez la section Publication des journaux Neptune sur Amazon CloudWatch Logs.

- Endpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le point de terminaison de connexion pour l'instance principale du cluster de bases de données.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour ce cluster de bases de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.

- HostedZoneId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'ID attribué par Amazon Route 53 lorsque vous créez une zone hébergée.

- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).

- IOOptimizedNextAllowedModificationTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  La prochaine fois, vous pourrez modifier le cluster de bases de données de façon à utiliser le type de stockage `iopt1`.
- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.
- LatestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée à l'aide de la fonction de point-in-time restauration.
- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de bases de données possède des instances dans plusieurs zones de disponibilité.
- PendingModifiedValues : objet ClusterPendingModifiedValues.

  Ce type de données est utilisé comme élément de réponse dans l'opération `ModifyDBCluster` et contient les modifications qui seront appliquées lors de la fenêtre de maintenance suivante.
- PercentProgress : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la progression de l'opération sous forme de pourcentage.
- Port— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le port sur lequel le moteur de base de données est à l'écoute.
- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.
- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.
- ReaderEndpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

Point de terminaison du lecteur pour le cluster de bases de données. Le point de terminaison du lecteur d'un cluster de bases de données équilibre la charge des connexions entre les réplicas en lecture qui sont disponibles dans un cluster de bases de données. À mesure que les clients demandent de nouvelles connexions au point de terminaison du lecteur, Neptune répartit les demandes de connexion entre les réplicas en lecture du cluster de bases de données. Cette fonctionnalité peut contribuer à équilibrer votre charge de travail entre les différents réplicas en lecture de votre cluster de bases de données.

Si un basculement se produit et que le réplica en lecture auquel vous êtes connecté est promu en instance principale, votre connexion est supprimée. Pour continuer à envoyer votre charge de travail de lecture à d'autres réplicas en lecture du cluster, vous pouvez alors vous reconnecter au point de terminaison du lecteur.

- ReadReplicaIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à ce cluster de bases de données.

- ReplicationSourceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ReplicationType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ServerlessV2ScalingConfiguration : objet Serverless V2 ScalingConfigurationInfo.

  Affiche la configuration de mise à l'échelle pour un cluster de bases de données Neptune sans serveur.

  Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'état actuel de ce cluster de base de données.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de base de données est chiffré.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

Type de stockage utilisé par le cluster de bases de données.

Valeurs valides :

- **`standard`**— (valeur par défaut) Fournit un stockage de base de données économique pour les applications dont l' I/O utilisation est modérée à faible.
- **`iopt1`**— Permet un stockage optimisé pour les E/S, conçu pour répondre aux besoins de I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O débit.

  Le stockage optimisé pour les E/S Neptune n'est disponible qu'à partir de la version 1.3.0.0 du moteur.

- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit la liste des groupes de sécurité VPC auxquels appartient le cluster de base de données.

Erreurs

- DBClusterAlreadyExistsFault
- InsufficientStorageClusterCapacityFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- Non valide VPCNetwork StateFault
- Non valide DBCluster StateFault
- Non valide DBSubnet GroupStateFault
- InvalidSubnet
- Non valide DBInstance StateFault
- DBClusterParameterGroupNotFoundFault
- KMSKeyNotAccessibleFault
- DBClusterNotFoundFault
- DBInstanceNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- GlobalClusterNotFoundFault

- InvalidGlobalClusterStateFault

Supprimer DBCluster (action)

Le nom de la AWS CLI pour cette API est :`delete-db-cluster`.

L'DBCluster action Supprimer supprime un cluster de base de données précédemment provisionné. Quand vous supprimez un cluster de base de données, toutes les sauvegardes automatiques de ce cluster de base de données sont supprimées et ne peuvent pas être récupérées. Les instantanés manuels du cluster de base de données spécifié ne sont pas supprimés.

Notez que le cluster de base de données ne peut pas être supprimé si la protection contre la suppression est activée. Pour le supprimer, vous devez d'abord définir son champ `DeletionProtection` sur `False`.

Demande

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du cluster de base de données à supprimer. Ce paramètre n'est pas sensible à la casse.

  Contraintes :

  - Doit correspondre à un DBCluster identifiant existant.
- FinalDBSnapshotIdentifier (dans la CLI : `--final-db-snapshot-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du nouvel instantané de cluster de base de données créé lorsque `SkipFinalSnapshot` a la valeur `false`.

  > **Note**
  >
  > Le fait de spécifier ce paramètre et également de définir le paramètre `SkipFinalShapshot` sur true génère une erreur.

  Contraintes :

  - Doit comporter entre 1 et 255 lettres, chiffres ou traits d'union.

  - Le premier caractère doit être une lettre
  - Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs
- SkipFinalSnapshot (dans la CLI : `--skip-final-snapshot`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Détermine si un instantané de cluster de base de données final est créé avant la suppression du cluster de base de données. Si `true` est spécifié, aucun instantané de cluster de base de données n'est créé. Si `false` est spécifié, un instantané de cluster de base de données est créé avant la suppression du cluster de base de données.

  > **Note**
  >
  > Vous devez spécifier un paramètre `FinalDBSnapshotIdentifier` si `SkipFinalSnapshot` a la valeur `false`.

  Par défaut : `false`

Réponse

Contient les détails d'un cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans the section called “Décrivez DBClusters”.

- AllocatedStorage— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  `AllocatedStorage` renvoie toujours la valeur 1, car la taille de stockage d'un cluster de bases de données Neptune n'est pas fixe ; elle s'ajuste automatiquement en fonction des besoins.
- AssociatedRoles : tableau d'objets DBClusterRôle.

  Fournit la liste des rôles Amazon Identity and Access Management (IAM) associés au cluster de bases de données. Les rôles IAM associés à un cluster de bases de données autorisent celui-ci à accéder aux autres services Amazon en votre nom.
- AutomaticRestartTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Heure à laquelle le cluster de bases de données sera automatiquement redémarré.

- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du cluster de base de données peuvent être créées.
- BacktrackConsumedChangeRecords— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.
- BacktrackWindow— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.
- BackupRetentionPeriod— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.
- Capacity— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Non pris en charge par Neptune.
- CloneGroupId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifie le groupe de clones auquel est associé le cluster de bases de données.
- ClusterCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).
- CopyTagsToSnapshot— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur *`true`*, les balises sont copiées dans n'importe quel instantané du cluster de bases de données créé.
- CrossAccountClone— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur `true`, le cluster de bases de données peut être cloné sur plusieurs comptes.
- DatabaseName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nom de la base de données initiale de ce cluster de bases de données qui a été fourni au moment de la création, dans la mesure où un nom a été spécifié au moment de créer le cluster

de bases de données. Ce même nom est renvoyé pendant la durée de vie du cluster de bases de données.

- DBClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du cluster de bases de données.

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de cluster de bases de données fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie un cluster de bases de données.

- DBClusterMembers : tableau d'objets DBClusterMembre.

  Fournit la liste des instances qui composent le cluster de bases de données.

- DBClusterParameterGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du groupe de paramètres de cluster de base de données pour le cluster de base de données.

- DbClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à la région Amazon du cluster de bases de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS du cluster de base de données est consultée.

- DBSubnetGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie les informations sur le groupe de sous-réseaux associé au cluster de bases de données, notamment le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée.

- EarliestBacktrackTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Non pris en charge par Neptune.

- EarliestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

Spécifie l'heure la plus proche à laquelle une base de données peut être restaurée par point-in-time restauration.

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux que ce cluster de base de données est configuré pour exporter vers CloudWatch Logs. Les types de journaux valides sont les suivants : `audit` (pour publier les journaux d'audit CloudWatch) et slowquery (pour publier les journaux de requêtes lentes). CloudWatch Consultez la section Publication des journaux Neptune sur Amazon CloudWatch Logs.
- Endpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le point de terminaison de connexion pour l'instance principale du cluster de bases de données.
- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour ce cluster de bases de données.
- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.
- HostedZoneId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'ID attribué par Amazon Route 53 lorsque vous créez une zone hébergée.
- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).
- IOOptimizedNextAllowedModificationTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  La prochaine fois, vous pourrez modifier le cluster de bases de données de façon à utiliser le type de stockage `iopt1`.

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.
- LatestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée à l'aide de la fonction de point-in-time restauration.
- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de bases de données possède des instances dans plusieurs zones de disponibilité.
- PendingModifiedValues : objet ClusterPendingModifiedValues.

  Ce type de données est utilisé comme élément de réponse dans l'opération `ModifyDBCluster` et contient les modifications qui seront appliquées lors de la fenêtre de maintenance suivante.
- PercentProgress : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la progression de l'opération sous forme de pourcentage.
- Port— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le port sur lequel le moteur de base de données est à l'écoute.
- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.
- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.
- ReaderEndpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Point de terminaison du lecteur pour le cluster de bases de données. Le point de terminaison du lecteur d'un cluster de bases de données équilibre la charge des connexions entre les réplicas en lecture qui sont disponibles dans un cluster de bases de données. À mesure que les clients demandent de nouvelles connexions au point de terminaison du lecteur, Neptune répartit les

demandes de connexion entre les réplicas en lecture du cluster de bases de données. Cette fonctionnalité peut contribuer à équilibrer votre charge de travail entre les différents réplicas en lecture de votre cluster de bases de données.

Si un basculement se produit et que le réplica en lecture auquel vous êtes connecté est promu en instance principale, votre connexion est supprimée. Pour continuer à envoyer votre charge de travail de lecture à d'autres réplicas en lecture du cluster, vous pouvez alors vous reconnecter au point de terminaison du lecteur.

- ReadReplicaIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à ce cluster de bases de données.

- ReplicationSourceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ReplicationType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ServerlessV2ScalingConfiguration : objet Serverless V2 ScalingConfigurationInfo.

  Affiche la configuration de mise à l'échelle pour un cluster de bases de données Neptune sans serveur.

  Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'état actuel de ce cluster de base de données.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de base de données est chiffré.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de stockage utilisé par le cluster de bases de données.

  Valeurs valides :

  - **`standard`**— (valeur par défaut) Fournit un stockage de base de données économique pour les applications dont l' I/O utilisation est modérée à faible.

- **`iopt1`**— Permet un stockage optimisé pour les E/S, conçu pour répondre aux besoins de I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O débit.

  Le stockage optimisé pour les E/S Neptune n'est disponible qu'à partir de la version 1.3.0.0 du moteur.
- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit la liste des groupes de sécurité VPC auxquels appartient le cluster de base de données.

Erreurs

- DBClusterNotFoundFault
- Non valide DBCluster StateFault
- DBClusterSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault
- Non valide DBCluster SnapshotStateFault

Modifier DBCluster (action)

Le nom de la AWS CLI pour cette API est :`modify-db-cluster`.

Permet de modifier un paramètre de cluster de base de données. Vous pouvez modifier un ou plusieurs paramètres de configuration de base de données en spécifiant ces paramètres et les nouvelles valeurs dans la demande.

Demande

- AllowMajorVersionUpgrade (dans la CLI : `--allow-major-version-upgrade`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur qui indique si les mises à niveau entre différentes version majeures sont autorisées.

  Contraintes : vous devez définir l' allow-major-version-upgradeindicateur lorsque vous fournissez un `EngineVersion` paramètre qui utilise une version majeure différente de la version actuelle du cluster de bases de données.

- ApplyImmediately (dans la CLI : `--apply-immediately`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur qui indique si les modifications au niveau de cette demande et toutes les modifications en attente sont appliquées de manière asynchrone dès que possible, indépendamment du paramètre `PreferredMaintenanceWindow` du cluster de base de données. Si ce paramètre a la valeur `false`, les modifications apportées au cluster de base de données sont appliquées pendant la fenêtre de maintenance suivante.

  Le paramètre `ApplyImmediately` affecte uniquement les valeurs `NewDBClusterIdentifier`. Si vous définissez la valeur du paramètre `ApplyImmediately` sur false, les modifications apportées à la valeur `NewDBClusterIdentifier` seront appliquées pendant la fenêtre de maintenance suivante. Toutes les autres modifications sont appliquées immédiatement, quelle que soit la valeur du paramètre `ApplyImmediately`.

  Par défaut : `false`

- BackupRetentionPeriod(dans la CLI :`--backup-retention-period`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre de jours de conservation des sauvegardes automatiques. Vous devez spécifier une valeur minimale de 1.

  Par défaut : 1

  Contraintes :

  - Doit être une valeur comprise entre 1 et 35

- CloudwatchLogsExportConfiguration (dans la CLI : `--cloudwatch-logs-export-configuration`) : objet CloudwatchLogsExportConfiguration.

  Le paramètre de configuration des types de journaux à activer pour l'exportation vers les CloudWatch journaux pour un cluster de base de données spécifique. Consultez la section Utilisation de la CLI pour publier les journaux d'audit de Neptune dans Logs. CloudWatch

- CopyTagsToSnapshot(dans la CLI :`--copy-tags-to-snapshot`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur *`true`*, les balises sont copiées dans n'importe quel instantané du cluster de bases de données créé.

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du cluster de bases de données en cours de modification. Ce paramètre n'est pas sensible à la casse.

  Contraintes :

  - Doit correspondre à l'identifiant d'un existant DBCluster.
- DBClusterParameterGroupName (dans la CLI : `--db-cluster-parameter-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de cluster de base de données à utiliser pour le cluster de base de données.
- DBInstanceParameterGroupName (dans la CLI : `--db-instance-parameter-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le nom du groupe de paramètres de base de données à appliquer à toutes les instances du cluster de base de données.

  > **Note**
  >
  > Lorsque vous appliquez un groupe de paramètres à l'aide de `DBInstanceParameterGroupName`, les modifications de paramètres ne seront pas appliquées lors de la fenêtre de maintenance suivante. Elles sont effectives immédiatement.

  Par défaut : le paramètre de nom existant

  Contraintes :

  - Le groupe de paramètres de la base de données doit faire partie de la même famille de groupes de paramètres de la base de données que la version de ce cluster de bases de données cible.
  - Le paramètre `DBInstanceParameterGroupName` n'est valide qu'en combinaison avec le paramètre `AllowMajorVersionUpgrade`.
- DeletionProtection(dans la CLI :`--deletion-protection`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

Valeur qui indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée. Par défaut, la protection contre la suppression est désactivée.

- EnableIAMDatabaseAuthentication(dans la CLI :`--enable-iam-database-authentication`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Valeur true pour activer le mappage des comptes Amazon Identity and Access Management (IAM) avec les comptes de base de données ; sinon, valeur false.

  Par défaut : `false`

- EngineVersion (dans la CLI : `--engine-version`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Numéro de version du moteur de base de données vers lequel vous souhaitez effectuer la mise à niveau. La modification de ce paramètre entraîne une interruption. La modification est appliquée pendant la fenêtre de maintenance suivante, sauf si le paramètre `ApplyImmediately` a la valeur true.

  Pour obtenir la liste des versions de moteur valides, consultez Versions du moteur pour Amazon Neptune, ou appelez the section called "Décrire DBEngine les versions".

- NewDBClusterIdentifier (dans la CLI : `--new-db-cluster-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nouvel identifiant du cluster de base de données lorsque celui-ci est renommé. Cette valeur est stockée sous la forme d'une chaîne en minuscules.

  Contraintes :

  - Doit contenir entre 1 et 63 lettres, chiffres ou traits d'union
  - Le premier caractère doit être une lettre
  - Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs

  Exemple : `my-cluster2`

- Port(dans la CLI :`--port`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Le numéro de port sur lequel le cluster DB accepte des connexions.

  Contraintes : La valeur doit être comprise dans la plage `1150-65535`

Valeur par défaut : port du cluster de bases de données d'origine.

- PreferredBackupWindow (dans la CLI : `--preferred-backup-window`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Plage de temps quotidienne au cours de laquelle les sauvegardes automatiques sont créées si cette fonctionnalité est activée via le paramètre `BackupRetentionPeriod`.

  Par défaut, une fenêtre de 30 minutes est sélectionnée de manière aléatoire sur un bloc horaire de 8 heures pour chaque région Amazon.

  Contraintes :

  - Doit être au format `hh24:mi-hh24:mi`.
  - Doit être exprimée en heure UTC (Universal Coordinated Time).
  - Ne doit pas être en conflit avec la fenêtre de maintenance préférée.
  - Doit être de 30 minutes minimum.
- PreferredMaintenanceWindow (dans la CLI : `--preferred-maintenance-window`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.

  Format : `ddd:hh24:mi-ddd:hh24:mi`

  Par défaut, une fenêtre de 30 minutes est sélectionnée de manière aléatoire dans un bloc de 8 heures pour chaque région Amazon, se produisant un jour choisi au hasard dans la semaine.

  Jours valides : Mon, Tue, Wed, Thu, Fri, Sat, Sun.

  Contraintes : fenêtre minimale de 30 minutes.
- ServerlessV2ScalingConfiguration (dans la CLI : `--serverless-v2-scaling-configuration`) : objet Serverless V2 ScalingConfiguration.

  Contient la configuration de mise à l'échelle d'un cluster de bases de données Neptune sans serveur.

  Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

- StorageType (dans la CLI : `--storage-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de stockage à associer au cluster de base de données.

  Valeurs valides :

  - **`standard`**— (valeur par défaut) Configure un stockage de base de données rentable pour les applications dont l' I/O utilisation est modérée à faible.
  - **`iopt1`**— Permet un stockage optimisé pour les E/S, conçu pour répondre aux besoins de I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O débit.

    Le stockage optimisé pour les E/S Neptune n'est disponible qu'à partir de la version 1.3.0.0 du moteur.
- VpcSecurityGroupIds (dans la CLI : `--vpc-security-group-ids`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des groupes de sécurité VPC auxquels le cluster de base de données appartiendra.

Réponse

Contient les détails d'un cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans the section called "Décrivez DBClusters".

- AllocatedStorage— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  `AllocatedStorage` renvoie toujours la valeur 1, car la taille de stockage d'un cluster de bases de données Neptune n'est pas fixe ; elle s'ajuste automatiquement en fonction des besoins.
- AssociatedRoles : tableau d'objets DBClusterRôle.

  Fournit la liste des rôles Amazon Identity and Access Management (IAM) associés au cluster de bases de données. Les rôles IAM associés à un cluster de bases de données autorisent celui-ci à accéder aux autres services Amazon en votre nom.
- AutomaticRestartTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Heure à laquelle le cluster de bases de données sera automatiquement redémarré.

- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du cluster de base de données peuvent être créées.

- BacktrackConsumedChangeRecords— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.

- BacktrackWindow— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.

- BackupRetentionPeriod— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.

- Capacity— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Non pris en charge par Neptune.

- CloneGroupId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifie le groupe de clones auquel est associé le cluster de bases de données.

- ClusterCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).

- CopyTagsToSnapshot— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur *`true`*, les balises sont copiées dans n'importe quel instantané du cluster de bases de données créé.

- CrossAccountClone— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur `true`, le cluster de bases de données peut être cloné sur plusieurs comptes.

- DatabaseName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nom de la base de données initiale de ce cluster de bases de données qui a été fourni au moment de la création, dans la mesure où un nom a été spécifié au moment de créer le cluster

de bases de données. Ce même nom est renvoyé pendant la durée de vie du cluster de bases de données.

- DBClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du cluster de bases de données.

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de cluster de bases de données fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie un cluster de bases de données.

- DBClusterMembers : tableau d'objets DBClusterMembre.

  Fournit la liste des instances qui composent le cluster de bases de données.

- DBClusterParameterGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du groupe de paramètres de cluster de base de données pour le cluster de base de données.

- DbClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à la région Amazon du cluster de bases de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS du cluster de base de données est consultée.

- DBSubnetGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie les informations sur le groupe de sous-réseaux associé au cluster de bases de données, notamment le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée.

- EarliestBacktrackTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Non pris en charge par Neptune.

- EarliestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

Spécifie l'heure la plus proche à laquelle une base de données peut être restaurée par point-in-time restauration.

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux que ce cluster de base de données est configuré pour exporter vers CloudWatch Logs. Les types de journaux valides sont les suivants : `audit` (pour publier les journaux d'audit CloudWatch) et slowquery (pour publier les journaux de requêtes lentes). CloudWatch Consultez la section Publication des journaux Neptune sur Amazon CloudWatch Logs.
- Endpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le point de terminaison de connexion pour l'instance principale du cluster de bases de données.
- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour ce cluster de bases de données.
- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.
- HostedZoneId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'ID attribué par Amazon Route 53 lorsque vous créez une zone hébergée.
- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).
- IOOptimizedNextAllowedModificationTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  La prochaine fois, vous pourrez modifier le cluster de bases de données de façon à utiliser le type de stockage `iopt1`.

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.
- LatestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée à l'aide de la fonction de point-in-time restauration.
- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de bases de données possède des instances dans plusieurs zones de disponibilité.
- PendingModifiedValues : objet ClusterPendingModifiedValues.

  Ce type de données est utilisé comme élément de réponse dans l'opération `ModifyDBCluster` et contient les modifications qui seront appliquées lors de la fenêtre de maintenance suivante.
- PercentProgress : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la progression de l'opération sous forme de pourcentage.
- Port— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le port sur lequel le moteur de base de données est à l'écoute.
- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.
- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.
- ReaderEndpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Point de terminaison du lecteur pour le cluster de bases de données. Le point de terminaison du lecteur d'un cluster de bases de données équilibre la charge des connexions entre les réplicas en lecture qui sont disponibles dans un cluster de bases de données. À mesure que les clients demandent de nouvelles connexions au point de terminaison du lecteur, Neptune répartit les

demandes de connexion entre les réplicas en lecture du cluster de bases de données. Cette fonctionnalité peut contribuer à équilibrer votre charge de travail entre les différents réplicas en lecture de votre cluster de bases de données.

Si un basculement se produit et que le réplica en lecture auquel vous êtes connecté est promu en instance principale, votre connexion est supprimée. Pour continuer à envoyer votre charge de travail de lecture à d'autres réplicas en lecture du cluster, vous pouvez alors vous reconnecter au point de terminaison du lecteur.

- ReadReplicaIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à ce cluster de bases de données.

- ReplicationSourceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ReplicationType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ServerlessV2ScalingConfiguration : objet Serverless V2 ScalingConfigurationInfo.

  Affiche la configuration de mise à l'échelle pour un cluster de bases de données Neptune sans serveur.

  Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'état actuel de ce cluster de base de données.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de base de données est chiffré.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de stockage utilisé par le cluster de bases de données.

  Valeurs valides :

  - **`standard`**— (valeur par défaut) Fournit un stockage de base de données économique pour les applications dont l' I/O utilisation est modérée à faible.

- **`iopt1`**— Permet un stockage optimisé pour les E/S, conçu pour répondre aux besoins de I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O débit.

  Le stockage optimisé pour les E/S Neptune n'est disponible qu'à partir de la version 1.3.0.0 du moteur.
- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit la liste des groupes de sécurité VPC auxquels appartient le cluster de base de données.

Erreurs

- DBClusterNotFoundFault
- Non valide DBCluster StateFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- Non valide VPCNetwork StateFault
- Non valide DBSubnet GroupStateFault
- InvalidSubnet
- DBClusterParameterGroupNotFoundFault
- Non valide DBSecurity GroupStateFault
- Non valide DBInstance StateFault
- DBClusterAlreadyExistsFault
- StorageTypeNotSupportedFault

Démarrer DBCluster (action)

Le nom de la AWS CLI pour cette API est :`start-db-cluster`.

Démarre un cluster de base de données Amazon Neptune qui a été arrêté à l'aide de la console Amazon, de la stop-db-cluster commande Amazon CLI ou de l'API StopDBCluster .

Demande

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

Identifiant du cluster de base de données Neptune à démarrer. Ce paramètre est stocké sous la forme d'une chaîne en lettres minuscules.

Réponse

Contient les détails d'un cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans the section called "Décrivez DBClusters".

- AllocatedStorage— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  `AllocatedStorage` renvoie toujours la valeur 1, car la taille de stockage d'un cluster de bases de données Neptune n'est pas fixe ; elle s'ajuste automatiquement en fonction des besoins.
- AssociatedRoles : tableau d'objets DBClusterRôle.

  Fournit la liste des rôles Amazon Identity and Access Management (IAM) associés au cluster de bases de données. Les rôles IAM associés à un cluster de bases de données autorisent celui-ci à accéder aux autres services Amazon en votre nom.
- AutomaticRestartTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Heure à laquelle le cluster de bases de données sera automatiquement redémarré.
- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du cluster de base de données peuvent être créées.
- BacktrackConsumedChangeRecords— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.
- BacktrackWindow— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.
- BackupRetentionPeriod— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.
- Capacity— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

Non pris en charge par Neptune.

- CloneGroupId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifie le groupe de clones auquel est associé le cluster de bases de données.

- ClusterCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).

- CopyTagsToSnapshot— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur *`true`*, les balises sont copiées dans n'importe quel instantané du cluster de bases de données créé.

- CrossAccountClone— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur `true`, le cluster de bases de données peut être cloné sur plusieurs comptes.

- DatabaseName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nom de la base de données initiale de ce cluster de bases de données qui a été fourni au moment de la création, dans la mesure où un nom a été spécifié au moment de créer le cluster de bases de données. Ce même nom est renvoyé pendant la durée de vie du cluster de bases de données.

- DBClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du cluster de bases de données.

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de cluster de bases de données fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie un cluster de bases de données.

- DBClusterMembers : tableau d'objets DBClusterMembre.

  Fournit la liste des instances qui composent le cluster de bases de données.

- DBClusterParameterGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

Spécifie le nom du groupe de paramètres de cluster de base de données pour le cluster de base de données.

- DbClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à la région Amazon du cluster de bases de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS du cluster de base de données est consultée.

- DBSubnetGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie les informations sur le groupe de sous-réseaux associé au cluster de bases de données, notamment le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée.

- EarliestBacktrackTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Non pris en charge par Neptune.

- EarliestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure la plus proche à laquelle une base de données peut être restaurée par point-in-time restauration.

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux que ce cluster de base de données est configuré pour exporter vers CloudWatch Logs. Les types de journaux valides sont les suivants : `audit` (pour publier les journaux d'audit CloudWatch) et slowquery (pour publier les journaux de requêtes lentes). CloudWatch Consultez la section Publication des journaux Neptune sur Amazon CloudWatch Logs.

- Endpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le point de terminaison de connexion pour l'instance principale du cluster de bases de données.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour ce cluster de bases de données.
- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.
- HostedZoneId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'ID attribué par Amazon Route 53 lorsque vous créez une zone hébergée.
- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).
- IOOptimizedNextAllowedModificationTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  La prochaine fois, vous pourrez modifier le cluster de bases de données de façon à utiliser le type de stockage `iopt1`.
- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.
- LatestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée à l'aide de la fonction de point-in-time restauration.
- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de bases de données possède des instances dans plusieurs zones de disponibilité.

- PendingModifiedValues : objet ClusterPendingModifiedValues.

  Ce type de données est utilisé comme élément de réponse dans l'opération `ModifyDBCluster` et contient les modifications qui seront appliquées lors de la fenêtre de maintenance suivante.
- PercentProgress : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la progression de l'opération sous forme de pourcentage.
- Port— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le port sur lequel le moteur de base de données est à l'écoute.
- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.
- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.
- ReaderEndpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Point de terminaison du lecteur pour le cluster de bases de données. Le point de terminaison du lecteur d'un cluster de bases de données équilibre la charge des connexions entre les réplicas en lecture qui sont disponibles dans un cluster de bases de données. À mesure que les clients demandent de nouvelles connexions au point de terminaison du lecteur, Neptune répartit les demandes de connexion entre les réplicas en lecture du cluster de bases de données. Cette fonctionnalité peut contribuer à équilibrer votre charge de travail entre les différents réplicas en lecture de votre cluster de bases de données.

  Si un basculement se produit et que le réplica en lecture auquel vous êtes connecté est promu en instance principale, votre connexion est supprimée. Pour continuer à envoyer votre charge de travail de lecture à d'autres réplicas en lecture du cluster, vous pouvez alors vous reconnecter au point de terminaison du lecteur.
- ReadReplicaIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à ce cluster de bases de données.
- ReplicationSourceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

Non pris en charge par Neptune.

- ReplicationType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ServerlessV2ScalingConfiguration : objet Serverless V2 ScalingConfigurationInfo.

  Affiche la configuration de mise à l'échelle pour un cluster de bases de données Neptune sans serveur.

  Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'état actuel de ce cluster de base de données.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de base de données est chiffré.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de stockage utilisé par le cluster de bases de données.

  Valeurs valides :

  - **`standard`**— (valeur par défaut) Fournit un stockage de base de données économique pour les applications dont l' I/O utilisation est modérée à faible.
  - **`iopt1`**— Permet un stockage optimisé pour les E/S, conçu pour répondre aux besoins de I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O débit.

    Le stockage optimisé pour les E/S Neptune n'est disponible qu'à partir de la version 1.3.0.0 du moteur.

- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit la liste des groupes de sécurité VPC auxquels appartient le cluster de base de données.

Erreurs

- DBClusterNotFoundFault

- Non valide DBCluster StateFault
- Non valide DBInstance StateFault

Arrêter DBCluster (action)

Le nom de la AWS CLI pour cette API est :`stop-db-cluster`.

Arrête un cluster de base de données Amazon Neptune. Lorsque vous arrêtez un cluster de base de données, Neptune conserve les métadonnées du cluster de base de données, y compris ses points de terminaison et ses groupes de paramètres de base de données.

Neptune conserve également les journaux de transactions afin que vous puissiez effectuer une point-in-time restauration si nécessaire.

Demande

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de cluster de base de données Neptune à arrêter. Ce paramètre est stocké sous la forme d'une chaîne en lettres minuscules.

Réponse

Contient les détails d'un cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans the section called "Décrivez DBClusters".

- AllocatedStorage— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  `AllocatedStorage` renvoie toujours la valeur 1, car la taille de stockage d'un cluster de bases de données Neptune n'est pas fixe ; elle s'ajuste automatiquement en fonction des besoins.
- AssociatedRoles : tableau d'objets DBClusterRôle.

  Fournit la liste des rôles Amazon Identity and Access Management (IAM) associés au cluster de bases de données. Les rôles IAM associés à un cluster de bases de données autorisent celui-ci à accéder aux autres services Amazon en votre nom.

- AutomaticRestartTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Heure à laquelle le cluster de bases de données sera automatiquement redémarré.
- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du cluster de base de données peuvent être créées.
- BacktrackConsumedChangeRecords— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.
- BacktrackWindow— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.
- BackupRetentionPeriod— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.
- Capacity— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Non pris en charge par Neptune.
- CloneGroupId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifie le groupe de clones auquel est associé le cluster de bases de données.
- ClusterCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).
- CopyTagsToSnapshot— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur *`true`*, les balises sont copiées dans n'importe quel instantané du cluster de bases de données créé.
- CrossAccountClone— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur `true`, le cluster de bases de données peut être cloné sur plusieurs comptes.

- DatabaseName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nom de la base de données initiale de ce cluster de bases de données qui a été fourni au moment de la création, dans la mesure où un nom a été spécifié au moment de créer le cluster de bases de données. Ce même nom est renvoyé pendant la durée de vie du cluster de bases de données.

- DBClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du cluster de bases de données.

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de cluster de bases de données fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie un cluster de bases de données.

- DBClusterMembers : tableau d'objets DBClusterMembre.

  Fournit la liste des instances qui composent le cluster de bases de données.

- DBClusterParameterGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du groupe de paramètres de cluster de base de données pour le cluster de base de données.

- DbClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à la région Amazon du cluster de bases de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS du cluster de base de données est consultée.

- DBSubnetGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie les informations sur le groupe de sous-réseaux associé au cluster de bases de données, notamment le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée.

- EarliestBacktrackTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

Non pris en charge par Neptune.

- EarliestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure la plus proche à laquelle une base de données peut être restaurée par point-in-time restauration.

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux que ce cluster de base de données est configuré pour exporter vers CloudWatch Logs. Les types de journaux valides sont les suivants : `audit` (pour publier les journaux d'audit CloudWatch) et slowquery (pour publier les journaux de requêtes lentes). CloudWatch Consultez la section Publication des journaux Neptune sur Amazon CloudWatch Logs.

- Endpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le point de terminaison de connexion pour l'instance principale du cluster de bases de données.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour ce cluster de bases de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.

- HostedZoneId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'ID attribué par Amazon Route 53 lorsque vous créez une zone hébergée.

- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).

- IOOptimizedNextAllowedModificationTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  La prochaine fois, vous pourrez modifier le cluster de bases de données de façon à utiliser le type de stockage `iopt1`.
- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.
- LatestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée à l'aide de la fonction de point-in-time restauration.
- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de bases de données possède des instances dans plusieurs zones de disponibilité.
- PendingModifiedValues : objet ClusterPendingModifiedValues.

  Ce type de données est utilisé comme élément de réponse dans l'opération `ModifyDBCluster` et contient les modifications qui seront appliquées lors de la fenêtre de maintenance suivante.
- PercentProgress : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la progression de l'opération sous forme de pourcentage.
- Port— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le port sur lequel le moteur de base de données est à l'écoute.
- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.
- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.
- ReaderEndpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

Point de terminaison du lecteur pour le cluster de bases de données. Le point de terminaison du lecteur d'un cluster de bases de données équilibre la charge des connexions entre les réplicas en lecture qui sont disponibles dans un cluster de bases de données. À mesure que les clients demandent de nouvelles connexions au point de terminaison du lecteur, Neptune répartit les demandes de connexion entre les réplicas en lecture du cluster de bases de données. Cette fonctionnalité peut contribuer à équilibrer votre charge de travail entre les différents réplicas en lecture de votre cluster de bases de données.

Si un basculement se produit et que le réplica en lecture auquel vous êtes connecté est promu en instance principale, votre connexion est supprimée. Pour continuer à envoyer votre charge de travail de lecture à d'autres réplicas en lecture du cluster, vous pouvez alors vous reconnecter au point de terminaison du lecteur.

- ReadReplicaIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à ce cluster de bases de données.

- ReplicationSourceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ReplicationType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ServerlessV2ScalingConfiguration : objet Serverless V2 ScalingConfigurationInfo.

  Affiche la configuration de mise à l'échelle pour un cluster de bases de données Neptune sans serveur.

  Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'état actuel de ce cluster de base de données.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de base de données est chiffré.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

Type de stockage utilisé par le cluster de bases de données.

Valeurs valides :

- **`standard`**— (valeur par défaut) Fournit un stockage de base de données économique pour les applications dont l' I/O utilisation est modérée à faible.
- **`iopt1`**— Permet un stockage optimisé pour les E/S, conçu pour répondre aux besoins de I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O débit.

  Le stockage optimisé pour les E/S Neptune n'est disponible qu'à partir de la version 1.3.0.0 du moteur.

- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit la liste des groupes de sécurité VPC auxquels appartient le cluster de base de données.

Erreurs

- DBClusterNotFoundFault
- Non valide DBCluster StateFault
- Non valide DBInstance StateFault

AddRoleToDBCluster (action)

Le nom de la AWS CLI pour cette API est :`add-role-to-db-cluster`.

Associe un rôle Identity and Access Management (IAM) à partir d'un cluster de bases de données Neptune.

Demande

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du cluster de base de données auquel associer le rôle IAM.

- FeatureName (dans la CLI : `--feature-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

Nom de la fonctionnalité de cluster de bases de données auquel le rôle IAM doit être associé. Pour obtenir la liste des noms de fonctions pris en charge, consultez the section called "DBEngineVersion".

- RoleArn (dans la CLI : `--role-arn`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du rôle IAM à associer au cluster de base de données, par exemple `arn:aws:iam::123456789012:role/NeptuneAccessRole`.

Réponse

- Paramètres d'absence de réponse.

Erreurs

- DBClusterNotFoundFault
- DBClusterRoleAlreadyExistsFault
- Non valide DBCluster StateFault
- DBClusterRoleQuotaExceededFault

RemoveRoleFromDBCluster (action)

Le nom de la AWS CLI pour cette API est :`remove-role-from-db-cluster`.

Dissocie un rôle Identity and Access Management (IAM) d'un cluster de base de données.

Demande

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du cluster de base de données à partir duquel le rôle IAM doit être dissocié.

- FeatureName (dans la CLI : `--feature-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

Nom de la fonctionnalité de cluster de bases de données dont le rôle IAM doit être dissocié. Pour obtenir la liste des noms de fonctions pris en charge, consultez the section called "Décrire DBEngine les versions".

- RoleArn (dans la CLI : `--role-arn`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du rôle IAM à dissocier du cluster de base de données, par exemple `arn:aws:iam::123456789012:role/NeptuneAccessRole`.

Réponse

- Paramètres d'absence de réponse.

Erreurs

- DBClusterNotFoundFault
- DBClusterRoleNotFoundFault
- Non valide DBCluster StateFault

Basculement DBCluster (action)

Le nom de la AWS CLI pour cette API est :`failover-db-cluster`.

Force un basculement pour un cluster de base de données.

Un basculement pour un cluster de bases de données promeut l'un des réplicas en lecture (instances en lecture seule) du cluster de bases de données en instance principale (enregistreur du cluster).

Amazon Neptune bascule automatiquement vers un réplica en lecture, s'il en existe un, lorsque l'instance principale échoue. Vous pouvez forcer un basculement lorsque vous souhaitez simuler un échec d'une instance principale à des fins de test. Étant donné que chaque instance d'un cluster de base de données possède sa propre adresse de point de terminaison, vous devez nettoyer et établir à nouveau toutes les connexions existantes qui utilisent ces adresses de point de terminaison lorsque le basculement est effectué.

Demande

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de cluster de base de données pour lequel forcer un basculement. Ce paramètre n'est pas sensible à la casse.

  Contraintes :

  - Doit correspondre à l'identifiant d'un existant DBCluster.
- TargetDBInstanceIdentifier (dans la CLI : `--target-db-instance-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de l'instance à promouvoir en instance principale.

  Vous devez spécifier l'identifiant d'instance pour un réplica en lecture du cluster de base de données. Par exemple, `mydbcluster-replica1`.

Réponse

Contient les détails d'un cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans the section called "Décrivez DBClusters".

- AllocatedStorage— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  `AllocatedStorage` renvoie toujours la valeur 1, car la taille de stockage d'un cluster de bases de données Neptune n'est pas fixe ; elle s'ajuste automatiquement en fonction des besoins.
- AssociatedRoles : tableau d'objets DBClusterRôle.

  Fournit la liste des rôles Amazon Identity and Access Management (IAM) associés au cluster de bases de données. Les rôles IAM associés à un cluster de bases de données autorisent celui-ci à accéder aux autres services Amazon en votre nom.
- AutomaticRestartTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Heure à laquelle le cluster de bases de données sera automatiquement redémarré.
- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du cluster de base de données peuvent être créées.

- BacktrackConsumedChangeRecords— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.

- BacktrackWindow— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.

- BackupRetentionPeriod— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.

- Capacity— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Non pris en charge par Neptune.

- CloneGroupId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifie le groupe de clones auquel est associé le cluster de bases de données.

- ClusterCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).

- CopyTagsToSnapshot— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur *`true`*, les balises sont copiées dans n'importe quel instantané du cluster de bases de données créé.

- CrossAccountClone— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur `true`, le cluster de bases de données peut être cloné sur plusieurs comptes.

- DatabaseName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nom de la base de données initiale de ce cluster de bases de données qui a été fourni au moment de la création, dans la mesure où un nom a été spécifié au moment de créer le cluster

de bases de données. Ce même nom est renvoyé pendant la durée de vie du cluster de bases de données.

- DBClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du cluster de bases de données.

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de cluster de bases de données fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie un cluster de bases de données.

- DBClusterMembers : tableau d'objets DBClusterMembre.

  Fournit la liste des instances qui composent le cluster de bases de données.

- DBClusterParameterGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du groupe de paramètres de cluster de base de données pour le cluster de base de données.

- DbClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à la région Amazon du cluster de bases de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS du cluster de base de données est consultée.

- DBSubnetGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie les informations sur le groupe de sous-réseaux associé au cluster de bases de données, notamment le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée.

- EarliestBacktrackTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Non pris en charge par Neptune.

- EarliestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

Spécifie l'heure la plus proche à laquelle une base de données peut être restaurée par point-in-time restauration.

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux que ce cluster de base de données est configuré pour exporter vers CloudWatch Logs. Les types de journaux valides sont les suivants : `audit` (pour publier les journaux d'audit CloudWatch) et slowquery (pour publier les journaux de requêtes lentes). CloudWatch Consultez la section Publication des journaux Neptune sur Amazon CloudWatch Logs.
- Endpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le point de terminaison de connexion pour l'instance principale du cluster de bases de données.
- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour ce cluster de bases de données.
- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.
- HostedZoneId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'ID attribué par Amazon Route 53 lorsque vous créez une zone hébergée.
- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).
- IOOptimizedNextAllowedModificationTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  La prochaine fois, vous pourrez modifier le cluster de bases de données de façon à utiliser le type de stockage `iopt1`.

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.
- LatestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée à l'aide de la fonction de point-in-time restauration.
- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de bases de données possède des instances dans plusieurs zones de disponibilité.
- PendingModifiedValues : objet ClusterPendingModifiedValues.

  Ce type de données est utilisé comme élément de réponse dans l'opération `ModifyDBCluster` et contient les modifications qui seront appliquées lors de la fenêtre de maintenance suivante.
- PercentProgress : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la progression de l'opération sous forme de pourcentage.
- Port— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le port sur lequel le moteur de base de données est à l'écoute.
- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.
- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.
- ReaderEndpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Point de terminaison du lecteur pour le cluster de bases de données. Le point de terminaison du lecteur d'un cluster de bases de données équilibre la charge des connexions entre les réplicas en lecture qui sont disponibles dans un cluster de bases de données. À mesure que les clients demandent de nouvelles connexions au point de terminaison du lecteur, Neptune répartit les

demandes de connexion entre les réplicas en lecture du cluster de bases de données. Cette fonctionnalité peut contribuer à équilibrer votre charge de travail entre les différents réplicas en lecture de votre cluster de bases de données.

Si un basculement se produit et que le réplica en lecture auquel vous êtes connecté est promu en instance principale, votre connexion est supprimée. Pour continuer à envoyer votre charge de travail de lecture à d'autres réplicas en lecture du cluster, vous pouvez alors vous reconnecter au point de terminaison du lecteur.

- ReadReplicaIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à ce cluster de bases de données.

- ReplicationSourceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ReplicationType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ServerlessV2ScalingConfiguration : objet Serverless V2 ScalingConfigurationInfo.

  Affiche la configuration de mise à l'échelle pour un cluster de bases de données Neptune sans serveur.

  Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'état actuel de ce cluster de base de données.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de base de données est chiffré.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de stockage utilisé par le cluster de bases de données.

  Valeurs valides :

  - **`standard`**— (valeur par défaut) Fournit un stockage de base de données économique pour les applications dont l' I/O utilisation est modérée à faible.

- **`iopt1`**— Permet un stockage optimisé pour les E/S, conçu pour répondre aux besoins de I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O débit.

  Le stockage optimisé pour les E/S Neptune n'est disponible qu'à partir de la version 1.3.0.0 du moteur.
- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit la liste des groupes de sécurité VPC auxquels appartient le cluster de base de données.

Erreurs

- DBClusterNotFoundFault
- Non valide DBCluster StateFault
- Non valide DBInstance StateFault

PromoteReadReplicaDBCluster (action)

Le nom de la AWS CLI pour cette API est :`promote-read-replica-db-cluster`.

Non pris en charge.

Demande

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge.

Réponse

Contient les détails d'un cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans the section called "Décrivez DBClusters".

- AllocatedStorage— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

`AllocatedStorage` renvoie toujours la valeur 1, car la taille de stockage d'un cluster de bases de données Neptune n'est pas fixe ; elle s'ajuste automatiquement en fonction des besoins.

- AssociatedRoles : tableau d'objets DBClusterRôle.

  Fournit la liste des rôles Amazon Identity and Access Management (IAM) associés au cluster de bases de données. Les rôles IAM associés à un cluster de bases de données autorisent celui-ci à accéder aux autres services Amazon en votre nom.
- AutomaticRestartTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Heure à laquelle le cluster de bases de données sera automatiquement redémarré.
- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du cluster de base de données peuvent être créées.
- BacktrackConsumedChangeRecords— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.
- BacktrackWindow— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.
- BackupRetentionPeriod— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.
- Capacity— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Non pris en charge par Neptune.
- CloneGroupId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifie le groupe de clones auquel est associé le cluster de bases de données.
- ClusterCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).
- CopyTagsToSnapshot— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

Si cette valeur est définie sur *`true`*, les balises sont copiées dans n'importe quel instantané du cluster de bases de données créé.

- CrossAccountClone— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur `true`, le cluster de bases de données peut être cloné sur plusieurs comptes.

- DatabaseName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nom de la base de données initiale de ce cluster de bases de données qui a été fourni au moment de la création, dans la mesure où un nom a été spécifié au moment de créer le cluster de bases de données. Ce même nom est renvoyé pendant la durée de vie du cluster de bases de données.

- DBClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du cluster de bases de données.

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de cluster de bases de données fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie un cluster de bases de données.

- DBClusterMembers : tableau d'objets DBClusterMembre.

  Fournit la liste des instances qui composent le cluster de bases de données.

- DBClusterParameterGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du groupe de paramètres de cluster de base de données pour le cluster de base de données.

- DbClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à la région Amazon du cluster de bases de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS du cluster de base de données est consultée.

- DBSubnetGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie les informations sur le groupe de sous-réseaux associé au cluster de bases de données, notamment le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée.

- EarliestBacktrackTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Non pris en charge par Neptune.

- EarliestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure la plus proche à laquelle une base de données peut être restaurée par point-in-time restauration.

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux que ce cluster de base de données est configuré pour exporter vers CloudWatch Logs. Les types de journaux valides sont les suivants : `audit` (pour publier les journaux d'audit CloudWatch) et slowquery (pour publier les journaux de requêtes lentes). CloudWatch Consultez la section Publication des journaux Neptune sur Amazon CloudWatch Logs.

- Endpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le point de terminaison de connexion pour l'instance principale du cluster de bases de données.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour ce cluster de bases de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.

- HostedZoneId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'ID attribué par Amazon Route 53 lorsque vous créez une zone hébergée.

- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).

- IOOptimizedNextAllowedModificationTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  La prochaine fois, vous pourrez modifier le cluster de bases de données de façon à utiliser le type de stockage `iopt1`.

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.

- LatestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée à l'aide de la fonction de point-in-time restauration.

- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de bases de données possède des instances dans plusieurs zones de disponibilité.

- PendingModifiedValues : objet ClusterPendingModifiedValues.

  Ce type de données est utilisé comme élément de réponse dans l'opération `ModifyDBCluster` et contient les modifications qui seront appliquées lors de la fenêtre de maintenance suivante.

- PercentProgress : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la progression de l'opération sous forme de pourcentage.

- Port— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le port sur lequel le moteur de base de données est à l'écoute.

- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.

- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.

- ReaderEndpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Point de terminaison du lecteur pour le cluster de bases de données. Le point de terminaison du lecteur d'un cluster de bases de données équilibre la charge des connexions entre les réplicas en lecture qui sont disponibles dans un cluster de bases de données. À mesure que les clients demandent de nouvelles connexions au point de terminaison du lecteur, Neptune répartit les demandes de connexion entre les réplicas en lecture du cluster de bases de données. Cette fonctionnalité peut contribuer à équilibrer votre charge de travail entre les différents réplicas en lecture de votre cluster de bases de données.

  Si un basculement se produit et que le réplica en lecture auquel vous êtes connecté est promu en instance principale, votre connexion est supprimée. Pour continuer à envoyer votre charge de travail de lecture à d'autres réplicas en lecture du cluster, vous pouvez alors vous reconnecter au point de terminaison du lecteur.

- ReadReplicaIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à ce cluster de bases de données.

- ReplicationSourceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ReplicationType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ServerlessV2ScalingConfiguration : objet Serverless V2 ScalingConfigurationInfo.

  Affiche la configuration de mise à l'échelle pour un cluster de bases de données Neptune sans serveur.

Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'état actuel de ce cluster de base de données.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de base de données est chiffré.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de stockage utilisé par le cluster de bases de données.

  Valeurs valides :

  - **`standard`**— (valeur par défaut) Fournit un stockage de base de données économique pour les applications dont l' I/O utilisation est modérée à faible.
  - **`iopt1`**— Permet un stockage optimisé pour les E/S, conçu pour répondre aux besoins de I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O débit.

    Le stockage optimisé pour les E/S Neptune n'est disponible qu'à partir de la version 1.3.0.0 du moteur.

- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit la liste des groupes de sécurité VPC auxquels appartient le cluster de base de données.

Erreurs

- DBClusterNotFoundFault
- Non valide DBCluster StateFault

Décrire DBClusters (action)

Le nom de la AWS CLI pour cette API est :`describe-db-clusters`.

Renvoie des informations sur les clusters de base de données provisionnés et prend en charge la pagination.

> **Note**
>
> Cette opération peut également renvoyer des informations pour les clusters Amazon RDS et les clusters Amazon DocDB.

Demande

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de cluster de base de données fourni par l'utilisateur. Si ce paramètre est spécifié, seules les informations concernant le cluster de base de données en question sont renvoyées. Ce paramètre n'est pas sensible à la casse.

  Contraintes :

  - S'il est fourni, il doit correspondre à un DBCluster identifiant existant.
- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Filtre qui spécifie un ou plusieurs clusters de base de données à décrire.

  Filtres pris en charge :

  - `db-cluster-id`- Accepte les identifiants de cluster de base de données et les Amazon Resource Names (ARNs) du cluster de bases de données. La liste des résultats inclura uniquement des informations sur les clusters de base de données identifiés par ceux-ci ARNs.
  - `engine` - Accepte un nom de moteur (tel que `neptune`) et restreint la liste de résultats aux clusters de base de données créés par ce moteur.

  Par exemple, pour invoquer cette API à partir de l'interface de ligne de commande Amazon et effectuer un filtrage afin que seuls les clusters de bases de données Neptune soient renvoyés, vous pouvez utiliser la commande suivante :

  Example

```
aws neptune describe-db-clusters \
            --filters  Name=engine,Values=neptune
```

- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

Jeton de pagination facultatif fourni par une demande the section called "Décrivez DBClusters" précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.

Réponse

- DBClusters : tableau d'objets DBCluster.

  Contient une liste de clusters de base de données pour l'utilisateur.
- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Un jeton de pagination qui peut être utilisé dans une DBClusters demande Describe ultérieure.

Erreurs

- DBClusterNotFoundFault

Structures :

DBCluster (structure)

Contient les détails d'un cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans the section called "Décrivez DBClusters".

Champs

- AllocatedStorage— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  `AllocatedStorage` renvoie toujours la valeur 1, car la taille de stockage d'un cluster de bases de données Neptune n'est pas fixe ; elle s'ajuste automatiquement en fonction des besoins.
- AssociatedRoles : tableau d'objets DBClusterRôle.

  Fournit la liste des rôles Amazon Identity and Access Management (IAM) associés au cluster de bases de données. Les rôles IAM associés à un cluster de bases de données autorisent celui-ci à accéder aux autres services Amazon en votre nom.
- AutomaticRestartTime— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Heure à laquelle le cluster de bases de données sera automatiquement redémarré.
- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du cluster de base de données peuvent être créées.
- BacktrackConsumedChangeRecords— Il s'agit d'un LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.
- BacktrackWindow— Il s'agit d'un LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.
- BackupRetentionPeriod— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.
- Capacity— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Non pris en charge par Neptune.
- CloneGroupId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifie le groupe de clones auquel est associé le cluster de bases de données.
- ClusterCreateTime— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).

- CopyTagsToSnapshot— Il s'agit d'une BooleanOptional valeur booléenne `boolean` (vraie ou fausse) du type : (valeur booléenne).

  Si cette valeur est définie sur *`true`*, les balises sont copiées dans n'importe quel instantané du cluster de bases de données créé.

- CrossAccountClone— Il s'agit d'une BooleanOptional valeur booléenne `boolean` (vraie ou fausse) du type : (valeur booléenne).

  Si cette valeur est définie sur `true`, le cluster de bases de données peut être cloné sur plusieurs comptes.

- DatabaseName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nom de la base de données initiale de ce cluster de bases de données qui a été fourni au moment de la création, dans la mesure où un nom a été spécifié au moment de créer le cluster de bases de données. Ce même nom est renvoyé pendant la durée de vie du cluster de bases de données.

- DBClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du cluster de bases de données.

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de cluster de bases de données fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie un cluster de bases de données.

- DBClusterMembers : tableau d'objets DBClusterMembre.

  Fournit la liste des instances qui composent le cluster de bases de données.

- DBClusterParameterGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du groupe de paramètres de cluster de base de données pour le cluster de base de données.

- DbClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à la région Amazon du cluster de bases de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS du cluster de base de données est consultée.

- DBSubnetGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie les informations sur le groupe de sous-réseaux associé au cluster de bases de données, notamment le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— Il s'agit d'une BooleanOptional valeur booléenne `boolean` (vraie ou fausse) du type : (valeur booléenne).

  Indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée.

- EarliestBacktrackTime— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Non pris en charge par Neptune.

- EarliestRestorableTime— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure la plus proche à laquelle une base de données peut être restaurée par point-in-time restauration.

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux que ce cluster de base de données est configuré pour exporter vers CloudWatch Logs. Les types de journaux valides sont les suivants : `audit` (pour publier les journaux d'audit CloudWatch) et slowquery (pour publier les journaux de requêtes lentes). CloudWatch Consultez la section Publication des journaux Neptune sur Amazon CloudWatch Logs.

- Endpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le point de terminaison de connexion pour l'instance principale du cluster de bases de données.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour ce cluster de bases de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.

- GlobalClusterIdentifier— Il s'agit d'un GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.
- HostedZoneId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'ID attribué par Amazon Route 53 lorsque vous créez une zone hébergée.
- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).
- IOOptimizedNextAllowedModificationTime— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  La prochaine fois, vous pourrez modifier le cluster de bases de données de façon à utiliser le type de stockage `iopt1`.
- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.
- LatestRestorableTime— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée à l'aide de la fonction de point-in-time restauration.
- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de bases de données possède des instances dans plusieurs zones de disponibilité.
- PendingModifiedValues : objet ClusterPendingModifiedValues.

  Ce type de données est utilisé comme élément de réponse dans l'opération `ModifyDBCluster` et contient les modifications qui seront appliquées lors de la fenêtre de maintenance suivante.
- PercentProgress : chaîne de type : `string` (chaîne encodée en UTF-8).

Spécifie la progression de l'opération sous forme de pourcentage.

- Port— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le port sur lequel le moteur de base de données est à l'écoute.

- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.

- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.

- ReaderEndpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Point de terminaison du lecteur pour le cluster de bases de données. Le point de terminaison du lecteur d'un cluster de bases de données équilibre la charge des connexions entre les réplicas en lecture qui sont disponibles dans un cluster de bases de données. À mesure que les clients demandent de nouvelles connexions au point de terminaison du lecteur, Neptune répartit les demandes de connexion entre les réplicas en lecture du cluster de bases de données. Cette fonctionnalité peut contribuer à équilibrer votre charge de travail entre les différents réplicas en lecture de votre cluster de bases de données.

  Si un basculement se produit et que le réplica en lecture auquel vous êtes connecté est promu en instance principale, votre connexion est supprimée. Pour continuer à envoyer votre charge de travail de lecture à d'autres réplicas en lecture du cluster, vous pouvez alors vous reconnecter au point de terminaison du lecteur.

- ReadReplicaIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à ce cluster de bases de données.

- ReplicationSourceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ReplicationType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ServerlessV2ScalingConfiguration : objet Serverless V2 ScalingConfigurationInfo.

  Affiche la configuration de mise à l'échelle pour un cluster de bases de données Neptune sans serveur.

  Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.
- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'état actuel de ce cluster de base de données.
- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de base de données est chiffré.
- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de stockage utilisé par le cluster de bases de données.

  Valeurs valides :

  - **`standard`**— (valeur par défaut) Fournit un stockage de base de données économique pour les applications dont l' I/O utilisation est modérée à faible.
  - **`iopt1`**— Permet un stockage optimisé pour les E/S, conçu pour répondre aux besoins de I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O débit.

    Le stockage optimisé pour les E/S Neptune n'est disponible qu'à partir de la version 1.3.0.0 du moteur.
- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit la liste des groupes de sécurité VPC auxquels appartient le cluster de base de données.

`DBCluster` est utilisé comme élément de réponse pour :

- Créez DBCluster
- Supprimer DBCluster
- Basculement DBCluster
- Modifier DBCluster
- PromoteReadReplicaDBCluster

- Restaurer DBCluster FromSnapshot
- Restaurer DBCluster ToPointInTime
- Démarrer DBCluster
- Arrêtez DBCluster

DBClusterMembre (structure)

Contient des informations sur une instance qui fait partie d'un cluster de base de données.

Champs

- DBClusterParameterGroupStatus : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'état du groupe de paramètres de cluster de base de données pour ce membre du cluster de base de données.
- DBInstanceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'identifiant d'instance de ce membre du cluster de base de données.
- IsClusterWriter : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur `true` si le membre du cluster est l'instance principale du cluster de base de données ; valeur `false` dans le cas contraire.
- PromotionTier— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Valeur qui spécifie l'ordre dans lequel un réplica en lecture est promu en instance principale après un échec de l'instance principale existante.

DBClusterRôle (structure)

Décrit un rôle Amazon Identity and Access Management (IAM) associé à un cluster de base de données.

Champs

- FeatureName : chaîne de type : `string` (chaîne encodée en UTF-8).

Le nom de la fonctionnalité associée au rôle Amazon Identity and Access Management (IAM). Pour obtenir la liste des noms de fonctions pris en charge, consultez the section called "Décrire DBEngine les versions".

- RoleArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'Amazon Resource Name (ARN) du rôle IAM associé au cluster de base de données.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Décrit l'état de l'association entre le rôle IAM et le cluster de base de données. La propriété Status renvoie l'une des valeurs suivantes :

  - `ACTIVE` : l'ARN du rôle IAM est associé au cluster de bases de données et peut être utilisé pour accéder à d'autres services Amazon en votre nom.
  - `PENDING` : l'ARN du rôle IAM est en cours d'association au cluster de bases de données.
  - `INVALID` : l'ARN du rôle IAM est associé au cluster de bases de données, mais le cluster de bases de données n'est pas en mesure d'endosser le rôle IAM pour accéder aux autres services Amazon en votre nom.

CloudwatchLogsExportConfiguration (structure)

Paramètre de configuration des types de journaux à activer pour l'exportation vers les CloudWatch journaux pour une instance de base de données ou un cluster de base de données spécifique.

Les `DisableLogTypes` tableaux `EnableLogTypes` et déterminent quels journaux seront exportés (ou non exportés) vers CloudWatch Logs.

Les types de journaux valides sont : `audit` (pour publier les journaux d'audit) et `slowquery` (pour publier les journaux de requêtes lentes). Consultez la section Publication des journaux Neptune sur Amazon CloudWatch Logs.

Champs

- DisableLogTypes : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux à désactiver.

- EnableLogTypes : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux à activer.

PendingCloudwatchLogsExports (structure)

Liste des types de journaux dont la configuration est toujours en attente. En d'autres termes, ces types de journaux sont en cours d'activation ou de désactivation.

Les types de journaux valides sont : `audit` (pour publier les journaux d'audit) et `slowquery` (pour publier les journaux de requêtes lentes). Consultez la section Publication des journaux Neptune sur Amazon CloudWatch Logs.

Champs

- LogTypesToDisable : chaîne de type : `string` (chaîne encodée en UTF-8).

  Types de journaux en cours d'activation. Une fois activés, ces types de journaux sont exportés vers CloudWatch Logs.
- LogTypesToEnable : chaîne de type : `string` (chaîne encodée en UTF-8).

  Types de journaux en cours de désactivation. Une fois désactivés, ces types de journaux ne sont pas exportés vers CloudWatch Logs.

ClusterPendingModifiedValues (structure)

Ce type de données est utilisé comme élément de réponse dans l'opération `ModifyDBCluster` et contient les modifications qui seront appliquées lors de la fenêtre de maintenance suivante.

Champs

- AllocatedStorage— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Taille de stockage allouée en gibioctets (Gio) pour les moteurs de bases de données. Pour Neptune, `AllocatedStorage` renvoie toujours la valeur 1, car la taille de stockage d'un cluster de bases de données Neptune n'est pas fixe. Elle s'ajuste automatiquement en fonction des besoins.
- BackupRetentionPeriod— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre de jours de conservation des instantanés de base de données automatiques.
- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  La valeur de l' DBClusteridentifiant du cluster de base de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Version du moteur de base de données.
- IAMDatabaseAuthenticationEnabled— Il s'agit d'une BooleanOptional valeur booléenne `boolean` (vraie ou fausse) du type : (valeur booléenne).

  Valeur qui indique si le mappage des comptes AWS Identity and Access Management (IAM) aux comptes de base de données est activé.
- Iops— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie la valeur des IOPS provisionnés (opérations d'E/S par seconde). Ce paramètre ne s'applique qu'aux clusters de bases de données Multi-AZ.
- PendingCloudwatchLogsExports : objet PendingCloudwatchLogsExports.

  Cette `PendingCloudwatchLogsExports` structure indique les modifications en attente concernant CloudWatch les journaux activés et ceux qui sont désactivés.
- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Modification en attente du type de stockage pour le cluster de bases de données. Valeurs valides :
  - **`standard`**— (valeur par défaut) Configure un stockage de base de données rentable pour les applications dont l' I/O utilisation est modérée à faible.
  - **`iopt1`**— Permet un stockage optimisé pour les E/S, conçu pour répondre aux besoins de I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O débit.

    Le stockage optimisé pour les E/S Neptune n'est disponible qu'à partir de la version 1.3.0.0 du moteur.

API de base de données globale Neptune

Actions :

- CreateGlobalCluster (action)
- DeleteGlobalCluster (action)
- ModifyGlobalCluster (action)
- DescribeGlobalClusters (action)

- FailoverGlobalCluster (action)
- RemoveFromGlobalCluster (action)

Structures :

- GlobalCluster (structure)
- GlobalClusterMember (structure)

CreateGlobalCluster (action)

Le nom de la AWS CLI pour cette API est :`create-global-cluster`.

Crée une base de données Neptune globale répartie sur plusieurs régions Amazon. La base de données globale contient un cluster principal unique doté d'une fonctionnalité de lecture-écriture, et des clusters secondaires en lecture seule qui reçoivent des données du cluster principal via une réplication haute vitesse effectuée par le sous-système de stockage Neptune.

Vous pouvez créer une base de données globale initialement vide, puis y ajouter un cluster principal et des clusters secondaires. Vous avez aussi la possibilité de spécifier un cluster Neptune existant pendant l'opération de création pour qu'il devienne le cluster principal de la base de données globale.

Demande

- DatabaseName (dans la CLI : `--database-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la nouvelle base de données globale (avec une limite maximale de 64 caractères alphanumériques).
- DeletionProtection(dans la CLI :`--deletion-protection`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Paramètre de protection contre la suppression pour la nouvelle base de données globale. La base de données globale ne peut pas être supprimée tant que la protection contre la suppression est activée.
- Engine (dans la CLI : `--engine`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du moteur de base de données à utiliser dans la base de données globale.

Valeurs valides : `neptune`

- EngineVersion (dans la CLI : `--engine-version`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Version de moteur Neptune à utiliser par la base de données globale.

  Valeurs valides : `1.2.0.0` ou plus.
- GlobalClusterIdentifier(dans la CLI :`--global-cluster-identifier`) — Obligatoire : a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Identifiant de cluster du nouveau cluster de bases de données globales.
- SourceDBClusterIdentifier (dans la CLI : `--source-db-cluster-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  (Facultatif) Amazon Resource Name (ARN) d'un cluster de bases de données Neptune existant à utiliser comme cluster principal de la nouvelle base de données globale.
- StorageEncrypted(dans la CLI :`--storage-encrypted`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Paramètre de chiffrement du stockage pour le nouveau cluster de bases de données globales.

Réponse

Contient les détails d'une base de données globale Amazon Neptune.

Ce type de données est utilisé comme élément de réponse pour les actions the section called "CreateGlobalCluster", the section called "DescribeGlobalClusters", the section called "ModifyGlobalCluster", the section called "DeleteGlobalCluster", the section called "FailoverGlobalCluster" et the section called "RemoveFromGlobalCluster".

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Paramètre de protection contre la suppression pour la base de données globale.
- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Moteur de base de données Neptune utilisé par la base de données globale (`"neptune"`).

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Version de moteur Neptune utilisée par la base de données globale.

- GlobalClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de la base de données globale.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.

- GlobalClusterMembers : tableau d'objets GlobalClusterMember.

  Une liste de clusters ARNs et d'instances ARNs pour tous les clusters de base de données qui font partie de la base de données globale.

- GlobalClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable de la base de données globale, unique dans toutes les régions. Cet identifiant se trouve dans les entrées du CloudTrail journal chaque fois que l'on accède à la clé KMS du cluster de base de données.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique l'état actuel de cette base de données globale.

- StorageEncrypted— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Paramètre de chiffrement du stockage pour la base de données globale.

Erreurs

- GlobalClusterAlreadyExistsFault
- GlobalClusterQuotaExceededFault
- Non valide DBCluster StateFault
- DBClusterNotFoundFault

DeleteGlobalCluster (action)

Le nom de la AWS CLI pour cette API est :`delete-global-cluster`.

Supprime une base de données globale. Le cluster principal et tous les clusters secondaires doivent d'abord être dissociés ou supprimés.

Demande

- GlobalClusterIdentifier(dans la CLI :`--global-cluster-identifier`) — Obligatoire : a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Identifiant du cluster de bases de données global qui est en cours de suppression.

Réponse

Contient les détails d'une base de données globale Amazon Neptune.

Ce type de données est utilisé comme élément de réponse pour les actions the section called "CreateGlobalCluster", the section called "DescribeGlobalClusters", the section called "ModifyGlobalCluster", the section called "DeleteGlobalCluster", the section called "FailoverGlobalCluster" et the section called "RemoveFromGlobalCluster".

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Paramètre de protection contre la suppression pour la base de données globale.
- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Moteur de base de données Neptune utilisé par la base de données globale (`"neptune"`).
- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Version de moteur Neptune utilisée par la base de données globale.
- GlobalClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de la base de données globale.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.

- GlobalClusterMembers : tableau d'objets GlobalClusterMember.

  Une liste de clusters ARNs et d'instances ARNs pour tous les clusters de base de données qui font partie de la base de données globale.

- GlobalClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable de la base de données globale, unique dans toutes les régions. Cet identifiant se trouve dans les entrées du CloudTrail journal chaque fois que l'on accède à la clé KMS du cluster de base de données.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique l'état actuel de cette base de données globale.

- StorageEncrypted— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Paramètre de chiffrement du stockage pour la base de données globale.

Erreurs

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

ModifyGlobalCluster (action)

Le nom de la AWS CLI pour cette API est :`modify-global-cluster`.

Modifie un paramètre pour un cluster global Amazon Neptune. Vous pouvez modifier un ou plusieurs paramètres de configuration de base de données en spécifiant ces paramètres et leurs nouvelles valeurs dans la demande.

Demande

- AllowMajorVersionUpgrade(dans la CLI :`--allow-major-version-upgrade`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

Une valeur qui indique que les mises à niveau de version majeures sont autorisées.

Contraintes : vous devez autoriser les mises à niveau de versions majeures si vous spécifiez une valeur pour le paramètre `EngineVersion` correspondant à une version majeure différente de la version actuelle du cluster de bases de données.

Si vous mettez à niveau la version majeure d'une base de données globale, les groupes de paramètres du cluster de bases de données et de l'instance de base de données sont définis sur les groupes de paramètres par défaut de la nouvelle version. Vous devrez donc appliquer tous les groupes de paramètres personnalisés une fois la mise à niveau terminée.

- DeletionProtection(dans la CLI :`--deletion-protection`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si la protection contre la suppression a été activée pour la base de données globale. La base de données globale ne peut pas être supprimée tant que la protection contre la suppression est activée.

- EngineVersion (dans la CLI : `--engine-version`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Numéro de version du moteur de base de données vers lequel vous souhaitez effectuer la mise à niveau. La modification de ce paramètre entraîne une interruption. La modification sera appliquée pendant la fenêtre de maintenance suivante, sauf si `ApplyImmediately` est activé.

  Pour répertorier toutes les versions disponibles du moteur Neptune, utilisez la commande suivante :

  Example

  ```
  aws neptune describe-db-engine-versions \
            --engine neptune \
            --query '*[]|[?SupportsGlobalDatabases == 'true'].[EngineVersion]'
  ```

- GlobalClusterIdentifier(dans la CLI :`--global-cluster-identifier`) — Obligatoire : a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Identifiant du cluster global de bases de données en cours de modification. Ce paramètre n'est pas sensible à la casse.

Contraintes : doit correspondre à l'identifiant d'un cluster de bases de données global existant.

- NewGlobalClusterIdentifier(dans la CLI :`--new-global-cluster-identifier`) — a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Nouvel identifiant de cluster à attribuer à la base de données globale. Cette valeur est stockée sous la forme d'une chaîne en minuscules.

  Contraintes :

  - Doit contenir entre 1 et 63 lettres, chiffres ou traits d'union.
  - Le premier caractère doit être une lettre.
  - Il ne peut pas se terminer par un trait d'union ou contenir deux traits d'union consécutifs.

  Exemple : `my-cluster2`

Réponse

Contient les détails d'une base de données globale Amazon Neptune.

Ce type de données est utilisé comme élément de réponse pour les actions the section called "CreateGlobalCluster", the section called "DescribeGlobalClusters", the section called "ModifyGlobalCluster", the section called "DeleteGlobalCluster", the section called "FailoverGlobalCluster" et the section called "RemoveFromGlobalCluster".

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Paramètre de protection contre la suppression pour la base de données globale.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Moteur de base de données Neptune utilisé par la base de données globale (`"neptune"`).

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Version de moteur Neptune utilisée par la base de données globale.

- GlobalClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de la base de données globale.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.
- GlobalClusterMembers : tableau d'objets GlobalClusterMember.

  Une liste de clusters ARNs et d'instances ARNs pour tous les clusters de base de données qui font partie de la base de données globale.
- GlobalClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable de la base de données globale, unique dans toutes les régions. Cet identifiant se trouve dans les entrées du CloudTrail journal chaque fois que l'on accède à la clé KMS du cluster de base de données.
- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique l'état actuel de cette base de données globale.
- StorageEncrypted— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Paramètre de chiffrement du stockage pour la base de données globale.

Erreurs

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

DescribeGlobalClusters (action)

> Le nom de la AWS CLI pour cette API est :`describe-global-clusters`.

Renvoie des informations sur les clusters de bases de données Neptune globales. Cette API prend en charge la pagination.

Demande

- GlobalClusterIdentifier(dans la CLI :`--global-cluster-identifier`) — a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Identifiant de cluster de base de données fourni par l'utilisateur. Si ce paramètre est spécifié, seules les informations concernant le cluster de bases de données en question sont renvoyées. Ce paramètre n'est pas sensible à la casse.

  Contraintes : si cette valeur est fournie, elle doit correspondre à l'identifiant d'un cluster de bases de données existant.
- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  (Facultatif) Jeton de pagination renvoyé par un appel précédent à `DescribeGlobalClusters`. Si ce paramètre est spécifié, la réponse inclut uniquement les enregistrements supérieurs au marqueur, jusqu'au nombre spécifié par `MaxRecords`.
- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de marqueur de pagination est inclus dans la réponse. Vous pouvez l'utiliser pour récupérer les résultats restants.

  Par défaut : `100`

  Contraintes : Minimum 20, maximum 100.

Réponse

- GlobalClusters : tableau d'objets GlobalCluster.

  Liste des instances et des clusters globaux renvoyés par cette demande.
- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination. Si ce paramètre est renvoyé dans la réponse, d'autres enregistrements sont disponibles. Ils peuvent être récupérés par un ou plusieurs appels supplémentaires à `DescribeGlobalClusters`.

Erreurs

- GlobalClusterNotFoundFault

FailoverGlobalCluster (action)

Le nom de la AWS CLI pour cette API est :`failover-global-cluster`.

Démarre le processus de basculement pour une base de données Neptune globale.

Un basculement d'une base de données globale Neptune promeut l'un des clusters de bases de données secondaires en lecture seule en cluster de bases de données principal et rétrograde le cluster de bases de données principal en cluster de bases de données secondaire (en lecture seule). En d'autres termes, le rôle du cluster de bases de données principal actuel et du cluster de bases de données secondaire cible sélectionné sont interchangés. Le cluster de base de données secondaire sélectionné suppose toutes read/write les fonctionnalités de la base de données globale Neptune.

> **Note**
>
> Cette action s'applique uniquement aux bases de données globales Neptune. Cette action vise uniquement à être utilisée sur les bases de données globales Neptune saines dotées de clusters de bases de données Neptune en bon état et sans interruption de service à l'échelle de la région. Elle permet de tester des scénarios de reprise après sinistre ou de reconfigurer la topologie de la base de données globale.

Demande

- GlobalClusterIdentifier(dans la CLI :`--global-cluster-identifier`) — Obligatoire : a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Identifiant de la base de données globale Neptune qui doit être basculée. L'identifiant est la clé unique attribuée par l'utilisateur lors de la création de la base de données globale Neptune. En d'autres termes, il s'agit du nom de la base de données globale pour laquelle vous souhaitez effectuer le basculement.

  Contraintes : doit correspondre à l'identifiant d'une base de données Neptune globale existante.

- TargetDbClusterIdentifier (dans la CLI : `--target-db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du cluster de bases de données Neptune secondaire que vous souhaitez promouvoir comme cluster principal pour la base de données globale.

Réponse

Contient les détails d'une base de données globale Amazon Neptune.

Ce type de données est utilisé comme élément de réponse pour les actions the section called "CreateGlobalCluster", the section called "DescribeGlobalClusters", the section called "ModifyGlobalCluster", the section called "DeleteGlobalCluster", the section called "FailoverGlobalCluster" et the section called "RemoveFromGlobalCluster".

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Paramètre de protection contre la suppression pour la base de données globale.
- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Moteur de base de données Neptune utilisé par la base de données globale (`"neptune"`).
- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Version de moteur Neptune utilisée par la base de données globale.
- GlobalClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de la base de données globale.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.
- GlobalClusterMembers : tableau d'objets GlobalClusterMember.

  Une liste de clusters ARNs et d'instances ARNs pour tous les clusters de base de données qui font partie de la base de données globale.

- GlobalClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable de la base de données globale, unique dans toutes les régions. Cet identifiant se trouve dans les entrées du CloudTrail journal chaque fois que l'on accède à la clé KMS du cluster de base de données.
- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique l'état actuel de cette base de données globale.
- StorageEncrypted— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Paramètre de chiffrement du stockage pour la base de données globale.

Erreurs

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault
- Non valide DBCluster StateFault
- DBClusterNotFoundFault

RemoveFromGlobalCluster (action)

Le nom de la AWS CLI pour cette API est :`remove-from-global-cluster`.

Dissocie un cluster de bases de données Neptune d'une base de données globale Neptune. Un cluster secondaire devient un cluster autonome normal doté d'une fonctionnalité de lecture-écriture au lieu d'être en lecture seule, et ne reçoit plus de données d'un cluster principal.

Demande

- DbClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) qui identifie le cluster à détacher du cluster de bases de données Neptune global.
- GlobalClusterIdentifier(dans la CLI :`--global-cluster-identifier`) — Obligatoire : a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

Identifiant de la base de données globale Neptune à partir de laquelle détacher le cluster de bases de données Neptune spécifié.

Réponse

Contient les détails d'une base de données globale Amazon Neptune.

Ce type de données est utilisé comme élément de réponse pour les actions the section called "CreateGlobalCluster", the section called "DescribeGlobalClusters", the section called "ModifyGlobalCluster", the section called "DeleteGlobalCluster", the section called "FailoverGlobalCluster" et the section called "RemoveFromGlobalCluster".

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Paramètre de protection contre la suppression pour la base de données globale.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Moteur de base de données Neptune utilisé par la base de données globale (`"neptune"`).

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Version de moteur Neptune utilisée par la base de données globale.

- GlobalClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de la base de données globale.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.

- GlobalClusterMembers : tableau d'objets GlobalClusterMember.

  Une liste de clusters ARNs et d'instances ARNs pour tous les clusters de base de données qui font partie de la base de données globale.

- GlobalClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

Identifiant immuable de la base de données globale, unique dans toutes les régions. Cet identifiant se trouve dans les entrées du CloudTrail journal chaque fois que l'on accède à la clé KMS du cluster de base de données.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique l'état actuel de cette base de données globale.

- StorageEncrypted— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Paramètre de chiffrement du stockage pour la base de données globale.

Erreurs

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault
- DBClusterNotFoundFault

Structures :

GlobalCluster (structure)

Contient les détails d'une base de données globale Amazon Neptune.

Ce type de données est utilisé comme élément de réponse pour les actions the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” et the section called “RemoveFromGlobalCluster”.

Champs

- DeletionProtection— Il s'agit d'une BooleanOptional valeur booléenne `boolean` (vraie ou fausse) du type : (valeur booléenne).

  Paramètre de protection contre la suppression pour la base de données globale.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Moteur de base de données Neptune utilisé par la base de données globale (`"neptune"`).

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Version de moteur Neptune utilisée par la base de données globale.

- GlobalClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de la base de données globale.

- GlobalClusterIdentifier— Il s'agit d'un GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.

- GlobalClusterMembers : tableau d'objets GlobalClusterMember.

  Une liste de clusters ARNs et d'instances ARNs pour tous les clusters de base de données qui font partie de la base de données globale.

- GlobalClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable de la base de données globale, unique dans toutes les régions. Cet identifiant se trouve dans les entrées du CloudTrail journal chaque fois que l'on accède à la clé KMS du cluster de base de données.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique l'état actuel de cette base de données globale.

- StorageEncrypted— Il s'agit d'une BooleanOptional valeur booléenne `boolean` (vraie ou fausse) du type : (valeur booléenne).

  Paramètre de chiffrement du stockage pour la base de données globale.

`GlobalCluster` est utilisé comme élément de réponse pour :

- CreateGlobalCluster
- ModifyGlobalCluster
- DeleteGlobalCluster
- RemoveFromGlobalCluster
- FailoverGlobalCluster

GlobalClusterMember (structure)

Structure de données contenant des informations sur les clusters principaux et secondaires associés à une base de données globale Neptune.

Champs

- DBClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du cluster de bases de données.
- IsWriter : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Spécifie si le cluster Neptune est le cluster principal (c'est-à-dire s'il possède une fonctionnalité de lecture-écriture) de la base de données globale Neptune à laquelle il est associé.
- Readers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de chaque cluster secondaire en lecture seule associé à la base de données Neptune globale.

API d'instances Neptune

Actions :

- Créer DBInstance (action)
- Supprimer DBInstance (action)
- Modifier DBInstance (action)
- Redémarrer DBInstance (action)
- Décrire DBInstances (action)
- DescribeOrderableDBInstanceOptions (action)
- DescribeValidDBInstanceModifications (action)

Structures :

- DBInstance (structure)
- DBInstanceStatusInfo (structure)
- DBInstanceOption commandable (structure)

- PendingModifiedValues (structure)
- ValidStorageOptions (structure)
- Valide DBInstance ModificationsMessage (structure)

Créer DBInstance (action)

Le nom de la AWS CLI pour cette API est :`create-db-instance`.

Crée une nouvelle instance de base de données.

Demande

- AutoMinorVersionUpgrade(dans la CLI :`--auto-minor-version-upgrade`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique que des mises à niveau de moteur mineures sont appliquées automatiquement à l'instance de base de données pendant la fenêtre de maintenance.

  Par défaut : `true`
- AvailabilityZone (dans la CLI : `--availability-zone`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  La zone de EC2 disponibilité dans laquelle l'instance de base de données est créée

  Valeur par défaut : zone de disponibilité aléatoire choisie par le système dans la région Amazon du point de terminaison.

  Exemple : `us-east-1d`

  Contrainte : le AvailabilityZone paramètre ne peut pas être spécifié si le paramètre MultiAZ est défini sur. `true` La zone de disponibilité spécifiée doit se trouver dans la même région Amazon que le point de terminaison actuel.
- BackupRetentionPeriod(dans la CLI :`--backup-retention-period`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre de jours de conservation des sauvegardes automatiques.

Non applicable. La période de rétention des sauvegardes automatisées est gérée par le cluster de bases de données. Pour de plus amples informations, veuillez consulter the section called "Créez DBCluster".

Par défaut : 1

Contraintes :

- Doit être compris entre 0 et 35
- Ne peut pas être défini sur 0 si l'instance de base de données est une source des réplicas en lecture

- CopyTagsToSnapshot(dans la CLI :`--copy-tags-to-snapshot`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Valeur true pour copier toutes les balises de l'instance de base de données vers les instantanés de l'instance de base de données, et false dans le cas contraire. La valeur par défaut est false.

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du cluster de bases de données auquel appartient l'instance.

  Pour plus d'informations sur la création d'un cluster de base de données , consultez the section called "Créez DBCluster".

  Type : String

- DBInstanceClass (dans la CLI : `--db-instance-class`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Capacité de calcul et de mémoire de l'instance de base de données, par exemple `db.m4.large`. Toutes les classes d'instances de base de données ne sont pas disponibles dans toutes les régions Amazon.

- DBInstanceIdentifier (dans la CLI : `--db-instance-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de l'instance de base de données. Ce paramètre est stocké sous la forme d'une chaîne en lettres minuscules.

  Contraintes :

- Doit contenir entre 1 et 63 lettres, chiffres ou traits d'union.
- Le premier caractère doit être une lettre.
- Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs.

Exemple : `mydbinstance`

- DBName (dans la CLI : `--db-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge.

- DBParameterGroupName (dans la CLI : `--db-parameter-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de base de données à associer à cette instance de base de données. Si cet argument est omis, le DBParameter groupe par défaut pour le moteur spécifié est utilisé.

  Contraintes :

  - Doit comporter entre 1 et 255 lettres, chiffres ou traits d'union.
  - Le premier caractère doit être une lettre
  - Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs

- DBSecurityGroups (dans la CLI : `--db-security-groups`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des Security Groups DB à associer à cette instance de base de données.

  Par défaut : le Security Group DB par défaut pour le moteur de base de données.

- DBSubnetGroupName (dans la CLI : `--db-subnet-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Groupe de sous-réseaux de base de données à associer à cette instance de base de données.

  S'il n'existe aucun groupe de sous-réseaux de base de données, il s'agit d'une instance de base de données non VPC.

- DeletionProtection(dans la CLI :`--deletion-protection`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Une valeur qui indique si la protection contre la suppression a été activée pour l'instance de base de données. La base de données ne peut pas être supprimée lorsque la protection contre la

suppression est activée. Par défaut, la protection contre la suppression est désactivée. Consultez Suppression d'une instance de base de données.

Les instances de base de données dans un cluster de base de données peuvent être supprimées même lorsque la protection contre la suppression est activée dans le cluster de base de données parent.

- Domain (dans la CLI : `--domain`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifiez le domaine Active Directory dans lequel créer l'instance.

- DomainIAMRoleName (dans la CLI : `--domain-iam-role-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifiez le nom du rôle IAM à utiliser pour effectuer des appels d'API à Directory Service.

- EnableCloudwatchLogsExports (dans la CLI : `--enable-cloudwatch-logs-exports`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux qui doivent être activés pour être exportés vers CloudWatch Logs.

- EnableIAMDatabaseAuthentication(dans la CLI :`--enable-iam-database-authentication`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Non prise en charge par Neptune (ignorée).

- Engine (dans la CLI : `--engine`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du moteur de base de données à utiliser pour cette instance.

  Valeurs valides : `neptune`

- EngineVersion (dans la CLI : `--engine-version`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Numéro de version du moteur de base de données à utiliser. Actuellement, la définition de ce paramètre n'a aucun effet.

- Iops(dans la CLI :`--iops`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Quantité d'IOPS provisionnés (opérations d'E/S par seconde) à allouer initialement pour l'instance de base de données.

- KmsKeyId (dans la CLI : `--kms-key-id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

Identifiant de clé Amazon KMS pour une instance de base de données chiffrée.

L'identifiant de clé KMS est l'Amazon Resource Name (ARN) de la clé de chiffrement KMS. Si vous créez une instance de base de données avec le compte Amazon qui détient la clé de chiffrement KMS utilisée pour chiffrer la nouvelle instance de base de données, vous pouvez utiliser l'alias de clé KMS plutôt que l'ARN de la clé de chiffrement KMS.

Non applicable. L'identifiant de clé KMS est géré par le cluster de bases de données. Pour de plus amples informations, veuillez consulter the section called “Créez DBCluster”.

Si le paramètre `StorageEncrypted` est true, et que vous ne spécifiez pas de valeur pour le paramètre `KmsKeyId`, Amazon Neptune utilisera votre clé de chiffrement par défaut. Amazon KMS crée la clé de chiffrement par défaut pour votre compte Amazon. Votre compte Amazon a une clé de chiffrement par défaut différente pour chaque région Amazon.

- LicenseModel (dans la CLI : `--license-model`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Informations sur le modèle de licence de cette instance de base de données.

  Valeurs valides : `license-included` | `bring-your-own-license` | `general-public-license`

- MonitoringInterval(dans la CLI :`--monitoring-interval`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Intervalle, en secondes, entre les points lorsque des métriques de surveillance améliorée sont collectées pour l'instance de base de données. Pour désactiver la collecte des métriques de surveillance améliorée, spécifiez 0. La valeur par défaut est 0.

  Si `MonitoringRoleArn` est spécifié, vous devez également définir `MonitoringInterval` sur une valeur différente de 0.

  Valeurs valides : `0, 1, 5, 10, 15, 30, 60`

- MonitoringRoleArn (dans la CLI : `--monitoring-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN du rôle IAM qui permet à Neptune d'envoyer des métriques de surveillance améliorées à Amazon CloudWatch Logs. Par exemple, `arn:aws:iam:123456789012:role/emaccess`.

Si `MonitoringInterval` est défini sur une valeur différente de 0, vous devez fournir une valeur `MonitoringRoleArn`.

- MultiAZ(dans la CLI :`--multi-az`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si l'instance de base de données est un déploiement multi-AZ. Vous ne pouvez pas définir le AvailabilityZone paramètre si le paramètre MultiAZ est défini sur true.

- Port(dans la CLI :`--port`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Numéro de port au niveau duquel la base de données accepte les connexions.

  Non applicable. Le port est géré par le cluster de bases de données. Pour de plus amples informations, veuillez consulter the section called "Créez DBCluster".

  Par défaut: `8182`

  Type : entier

- PreferredBackupWindow (dans la CLI : `--preferred-backup-window`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'intervalle de temps quotidien au cours duquel les sauvegardes automatiques sont créées.

  Non applicable. L'intervalle de temps quotidien pour la création des sauvegardes automatisées est géré par le cluster de bases de données. Pour de plus amples informations, veuillez consulter the section called "Créez DBCluster".

- PreferredMaintenanceWindow (dans la CLI : `--preferred-maintenance-window`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Intervalle de temps hebdomadaire, au format UTC (temps universel), pendant lequel a lieu la maintenance du système.

  Format : `ddd:hh24:mi-ddd:hh24:mi`

  Par défaut, une fenêtre de 30 minutes est sélectionnée de manière aléatoire dans un bloc de 8 heures pour chaque région Amazon, se produisant un jour choisi au hasard dans la semaine.

  Jours valides : Mon, Tue, Wed, Thu, Fri, Sat, Sun.

  Contraintes : fenêtre minimale de 30 minutes.

- PromotionTier(dans la CLI :`--promotion-tier`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Valeur qui spécifie l'ordre dans lequel un réplica en lecture est promu en instance principale après un échec de l'instance principale existante.

  Valeur par défaut : 1

  Valeurs valides : 0 - 15
- PubliclyAccessible(dans la CLI :`--publicly-accessible`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Cet indicateur ne doit plus être utilisé.
- StorageEncrypted(dans la CLI :`--storage-encrypted`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si l'instance de base de données est chiffrée.

  Non applicable. Le chiffrement des instances de base de données est géré par le cluster de bases de données. Pour de plus amples informations, veuillez consulter the section called "Créez DBCluster".

  Valeur par défaut : false
- StorageType (dans la CLI : `--storage-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non applicable. Dans Neptune, le type de stockage est géré au niveau du cluster de bases de données.
- Tags (dans la CLI : `--tags`) : tableau d'objets Tag.

  Balises à attribuer à la nouvelle instance.
- TdeCredentialArn (dans la CLI : `--tde-credential-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN du magasin de clés auquel associer l'instance pour le chiffrement TDE.
- TdeCredentialPassword(dans la CLI :`--tde-credential-password`) — a SensitiveString, de type : `string` (une chaîne codée en UTF-8).

  Mot de passe de l'ARN donnée du magasin de clés pour accéder à l'appareil.

- Timezone (dans la CLI : `--timezone`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fuseau horaire de l'instance de base de données.

- VpcSecurityGroupIds (dans la CLI : `--vpc-security-group-ids`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des groupes de sécurité EC2 VPC à associer à cette instance de base de données.

  Non applicable. La liste associée des groupes de sécurité EC2 VPC est gérée par le cluster de base de données. Pour de plus amples informations, veuillez consulter the section called “Créez DBCluster”.

  Par défaut : groupe de sécurité EC2 VPC par défaut pour le VPC du groupe de sous-réseaux de base de données.

Réponse

Contient les détails d'une instance de base de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called “Décrivez DBInstances”.

- AutoMinorVersionUpgrade : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique que des correctifs de versions mineures sont appliquées automatiquement.

- AvailabilityZone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom de la zone de disponibilité dans laquelle l'instance DB se trouve.

- BackupRetentionPeriod : entier de type : `integer` (entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.

- CACertificateIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du certificat CA de cette instance de base de données.

- CopyTagsToSnapshot : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si des balises sont copiées de l'instance de base de données vers des instantanés de l'instance de base de données.

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si l'instance de base de données est membre d'un cluster de bases de données, elle contient le nom du cluster de bases de données dont elle est membre.

- DBInstanceArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'instance de base de données.

- DBInstanceClass : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nom de la classe de capacité de calcul et de mémoire de l'instance de base de données.

- DBInstanceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de base de données fourni par l'utilisateur. Ce dernier est la clé unique qui identifie une instance de base de données.

- DbInstancePort : entier de type : `integer` (entier signé de 32 bits).

  Spécifie le port sur lequel l'instance de base de données écoute. Si l'instance de base de données fait partie d'un cluster de bases de données, le port peut être différent du port de cluster de bases de données.

- DBInstanceStatus : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique l'état actuel de cette base de données.

- DbiResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à chaque région Amazon pour l'instance de base de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS de l'instance de base de données est consultée.

- DBName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la base de données.

- DBParameterGroups : tableau d'objets DBParameterGroupStatus.

  Fournit la liste des groupes de paramètres de base de données appliqués à cette instance de base de données.

- DBSecurityGroups : tableau d'objets DBSecurityGroupMembership.

Fournit la liste des éléments du Security Group DB contenant uniquement des sous-éléments `DBSecurityGroup.Name` et `DBSecurityGroup.Status`.

- DBSubnetGroup : objet DBSubnetGroupe.

  Spécifie des informations sur le groupe de sous-réseaux associé à l'instance de base de données, dont le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si la protection contre la suppression a été activée pour l'instance de base de données. L'instance ne peut pas être supprimée tant que la protection contre la suppression est activée. Consultez Suppression d'une instance de base de données.

- DomainMemberships : tableau d'objets DomainMembership.

  Non pris en charge

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux que cette instance de base de données est configurée pour exporter vers CloudWatch Logs.

- Endpoint : objet Point de terminaison.

  Spécifie le point de terminaison de la connexion.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour cette instance de base de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.

- EnhancedMonitoringResourceArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le nom de ressource Amazon (ARN) du flux de journal Amazon CloudWatch Logs qui reçoit les données de métriques de surveillance améliorée pour l'instance de base de données.

- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true si l'authentification Amazon Identity and Access Management (IAM) est activée, et false dans le cas contraire.

- InstanceCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Indique la date et l'heure de création de l'instance de base de données.

- Iops— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie la valeur des IOPS provisionnés (opérations d'E/S par seconde).

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge : Le chiffrement des instances de base de données est géré par le cluster de bases de données.

- LatestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée par point-in-time restauration.

- LicenseModel : chaîne de type : `string` (chaîne encodée en UTF-8).

  Informations sur le modèle de licence de cette instance de base de données.

- MonitoringInterval— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Intervalle, en secondes, entre les points lorsque des métriques de surveillance améliorée sont collectées pour l'instance de base de données.

- MonitoringRoleArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN du rôle IAM qui permet à Neptune d'envoyer des métriques de surveillance améliorée à Amazon CloudWatch Logs.

- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si l'instance de base de données est un déploiement multi-AZ.

- PendingModifiedValues : objet PendingModifiedValues.

  Spécifie que les modifications apportées à l'instance de base de données sont en attente. Cet élément est inclus uniquement lorsque des modifications sont en attente. Des modifications spécifiques sont identifiées par sous-éléments.

- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.

- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.
- PromotionTier— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Valeur qui spécifie l'ordre dans lequel un réplica en lecture est promu en instance principale après un échec de l'instance principale existante.
- PubliclyAccessible : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Cet indicateur ne doit plus être utilisé.
- ReadReplicaDBClusterIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des clusters de bases de données qui sont des réplicas en lecture de cette instance de base de données.
- ReadReplicaDBInstanceIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à cette instance de base de données.
- ReadReplicaSourceDBInstanceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient l'identifiant de l'instance de base de données source si cette dernière est un réplica en lecture.
- SecondaryAvailabilityZone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si cette valeur est présente, spécifie le nom de la zone de disponibilité secondaire d'une instance de base de données avec prise en charge multi-AZ.
- StatusInfos : tableau d'objets DBInstanceStatusInfo.

  Statut d'un réplica en lecture. Si l'instance n'est pas un réplica en lecture, ce champ est vide.
- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Non pris en charge : Le chiffrement des instances de base de données est géré par le cluster de bases de données.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le type de stockage associé à l'instance de base de données.
- TdeCredentialArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN du magasin de clés associé à l'instance pour le chiffrement TDE.
- Timezone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge.
- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit une liste des éléments du groupe de sécurité VPC auxquels appartient l'instance de base de données.

Erreurs

- DBInstanceAlreadyExistsFault
- Insuffisant DBInstance CapacityFault
- DBParameterGroupNotFoundFault
- DBSecurityGroupNotFoundFault
- InstanceQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- Non valide DBCluster StateFault
- InvalidSubnet
- Non valide VPCNetwork StateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBClusterNotFoundFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- KMSKeyNotAccessibleFault

- DomainNotFoundFault

Supprimer DBInstance (action)

Le nom de la AWS CLI pour cette API est :`delete-db-instance`.

L'DBInstance action Supprimer supprime une instance de base de données précédemment provisionnée. Lorsque vous supprimez une instance de base de données, toutes les sauvegardes automatiques de cette dernière sont supprimées et ne peuvent pas être récupérées. Les instantanés de base de données manuels de l'instance de base de données devant être supprimés par l'action `DeleteDBInstance` ne sont pas supprimés.

Si vous demandez un instantané de bases de données final, le statut de l'instance de base de données Amazon Neptune est `deleting` jusqu'à la création de l'instantané de base de données. L'action d'API `DescribeDBInstance` est utilisée pour surveiller le statut de cette opération. L'action ne peut pas être annulée une fois exécutée.

Notez que lorsqu'une instance de base de données est dans un état d'échec et possède un statut `failed`, `incompatible-restore` ou `incompatible-network`, vous pouvez uniquement la supprimer lorsque le paramètre `SkipFinalSnapshot` est défini sur `true`.

Vous ne pouvez pas supprimer une instance de base de données si elle est la seule instance du cluster de base de données ou si la protection contre la suppression est activée.

Demande

- DBInstanceIdentifier (dans la CLI : `--db-instance-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant d'instance DB de l'instance de base de données à supprimer. Ce paramètre n'est pas sensible à la casse.

  Contraintes :

  - Doit correspondre au nom d'une instance de bases de données existante.
- FinalDBSnapshotIdentifier (dans la CLI : `--final-db-snapshot-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L' DBSnapshotidentifiant de la nouvelle date DBSnapshot créée SkipFinalSnapshot est défini sur`false`.

> **Note**
>
> Le fait de spécifier ce paramètre et de le SkipFinalShapshot définir également sur true entraîne une erreur.

Contraintes :

- Doit comporter entre 1 et 255 lettres ou chiffres.
- Le premier caractère doit être une lettre
- Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs
- Ne peut pas être spécifié lors de la suppression d'un réplica en lecture.

- SkipFinalSnapshot (dans la CLI : `--skip-final-snapshot`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Détermine si un instantané de bases de données final est créé avant la suppression de l'instance de base de données. S'il `true` est spécifié, aucun n' DBSnapshot est créé. Si `false` est spécifié, un instantané de base de données est créé avant que l'instance de base de données soit supprimée.

  Notez que lorsqu'une instance de base de données est en état de défaillance et a le statut « failed », « incompatible-restore » ou « incompatible-network », elle ne peut être supprimée que lorsque le SkipFinalSnapshot paramètre est défini sur « true ».

  Spécifiez `true` lors de la suppression d'un réplica en lecture.

  > **Note**
  >
  > Le paramètre Final DBSnapshot Identifier doit être spécifié si tel SkipFinalSnapshot est le cas`false`.

  Par défaut : `false`

Réponse

Contient les détails d'une instance de base de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called "Décrivez DBInstances".

- AutoMinorVersionUpgrade : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique que des correctifs de versions mineures sont appliquées automatiquement.
- AvailabilityZone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom de la zone de disponibilité dans laquelle l'instance DB se trouve.
- BackupRetentionPeriod : entier de type : `integer` (entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.
- CACertificateIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du certificat CA de cette instance de base de données.
- CopyTagsToSnapshot : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si des balises sont copiées de l'instance de base de données vers des instantanés de l'instance de base de données.
- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si l'instance de base de données est membre d'un cluster de bases de données, elle contient le nom du cluster de bases de données dont elle est membre.
- DBInstanceArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'instance de base de données.
- DBInstanceClass : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nom de la classe de capacité de calcul et de mémoire de l'instance de base de données.
- DBInstanceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de base de données fourni par l'utilisateur. Ce dernier est la clé unique qui identifie une instance de base de données.
- DbInstancePort : entier de type : `integer` (entier signé de 32 bits).

Spécifie le port sur lequel l'instance de base de données écoute. Si l'instance de base de données fait partie d'un cluster de bases de données, le port peut être différent du port de cluster de bases de données.

- DBInstanceStatus : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique l'état actuel de cette base de données.

- DbiResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à chaque région Amazon pour l'instance de base de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS de l'instance de base de données est consultée.

- DBName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la base de données.

- DBParameterGroups : tableau d'objets DBParameterGroupStatus.

  Fournit la liste des groupes de paramètres de base de données appliqués à cette instance de base de données.

- DBSecurityGroups : tableau d'objets DBSecurityGroupMembership.

  Fournit la liste des éléments du Security Group DB contenant uniquement des sous-éléments `DBSecurityGroup.Name` et `DBSecurityGroup.Status`.

- DBSubnetGroup : objet DBSubnetGroupe.

  Spécifie des informations sur le groupe de sous-réseaux associé à l'instance de base de données, dont le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si la protection contre la suppression a été activée pour l'instance de base de données. L'instance ne peut pas être supprimée tant que la protection contre la suppression est activée. Consultez Suppression d'une instance de base de données.

- DomainMemberships : tableau d'objets DomainMembership.

  Non pris en charge

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

Liste des types de journaux que cette instance de base de données est configurée pour exporter vers CloudWatch Logs.

- Endpoint : objet Point de terminaison.

  Spécifie le point de terminaison de la connexion.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour cette instance de base de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.

- EnhancedMonitoringResourceArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le nom de ressource Amazon (ARN) du flux de journal Amazon CloudWatch Logs qui reçoit les données de métriques de surveillance améliorée pour l'instance de base de données.

- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true si l'authentification Amazon Identity and Access Management (IAM) est activée, et false dans le cas contraire.

- InstanceCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Indique la date et l'heure de création de l'instance de base de données.

- Iops— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie la valeur des IOPS provisionnés (opérations d'E/S par seconde).

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge : Le chiffrement des instances de base de données est géré par le cluster de bases de données.

- LatestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée par point-in-time restauration.

- LicenseModel : chaîne de type : `string` (chaîne encodée en UTF-8).

Informations sur le modèle de licence de cette instance de base de données.

- MonitoringInterval— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Intervalle, en secondes, entre les points lorsque des métriques de surveillance améliorée sont collectées pour l'instance de base de données.

- MonitoringRoleArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN du rôle IAM qui permet à Neptune d'envoyer des métriques de surveillance améliorée à Amazon CloudWatch Logs.

- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si l'instance de base de données est un déploiement multi-AZ.

- PendingModifiedValues : objet PendingModifiedValues.

  Spécifie que les modifications apportées à l'instance de base de données sont en attente. Cet élément est inclus uniquement lorsque des modifications sont en attente. Des modifications spécifiques sont identifiées par sous-éléments.

- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.

- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.

- PromotionTier— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Valeur qui spécifie l'ordre dans lequel un réplica en lecture est promu en instance principale après un échec de l'instance principale existante.

- PubliclyAccessible : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Cet indicateur ne doit plus être utilisé.

- ReadReplicaDBClusterIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des clusters de bases de données qui sont des réplicas en lecture de cette instance de base de données.

- ReadReplicaDBInstanceIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à cette instance de base de données.
- ReadReplicaSourceDBInstanceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient l'identifiant de l'instance de base de données source si cette dernière est un réplica en lecture.
- SecondaryAvailabilityZone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si cette valeur est présente, spécifie le nom de la zone de disponibilité secondaire d'une instance de base de données avec prise en charge multi-AZ.
- StatusInfos : tableau d'objets DBInstanceStatusInfo.

  Statut d'un réplica en lecture. Si l'instance n'est pas un réplica en lecture, ce champ est vide.
- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Non pris en charge : Le chiffrement des instances de base de données est géré par le cluster de bases de données.
- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le type de stockage associé à l'instance de base de données.
- TdeCredentialArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN du magasin de clés associé à l'instance pour le chiffrement TDE.
- Timezone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge.
- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit une liste des éléments du groupe de sécurité VPC auxquels appartient l'instance de base de données.

Erreurs

- DBInstanceNotFoundFault
- Non valide DBInstance StateFault

- DBSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault
- Non valide DBCluster StateFault

Modifier DBInstance (action)

Le nom de la AWS CLI pour cette API est :`modify-db-instance`.

Modifie les paramètres d'une instance de base de données. Vous pouvez modifier un ou plusieurs paramètres de configuration de base de données en spécifiant ces paramètres et les nouvelles valeurs dans la demande. Pour en savoir plus sur les modifications à apporter à votre instance de base de données, appelez the section called "DescribeValidDBInstanceModifications" avant d'appeler the section called "Modifier DBInstance".

Demande

- AllowMajorVersionUpgrade (dans la CLI : `--allow-major-version-upgrade`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique que des mises à niveau de version majeure sont autorisées. La modification de ce paramètre n'entraîne pas d'interruption et est appliquée de manière asynchrone dès que possible.
- ApplyImmediately (dans la CLI : `--apply-immediately`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Spécifie si les modifications dans cette demande et toutes les modifications en attente sont appliquées de manière asynchrone dès que possible, quel que soit le paramètre `PreferredMaintenanceWindow` pour l'instance de base de données.

  Si ce paramètre est défini sur `false`, les modifications apportées à l'instance de base de données sont appliquées pendant la fenêtre de maintenance suivante. Certaines modifications de paramètre peut entraîner une interruption et sont appliquées sur le prochain appel à the section called "Redémarrer DBInstance", ou le prochain redémarrage d'échec.

  Par défaut : `false`
- AutoMinorVersionUpgrade(dans la CLI :`--auto-minor-version-upgrade`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

Indique que des mises à niveau de version mineure sont appliquées automatiquement à l'instance de base de données pendant la fenêtre de maintenance. La modification de ce paramètre n'entraîne pas d'interruption, sauf dans le cas suivant, et est appliquée de manière asynchrone dès que possible. Une interruption se produit si ce paramètre est défini sur `true` pendant la fenêtre de maintenance, si une version mineure plus récente est disponible et si Neptune active les correctifs automatiques pour cette version de moteur.

- BackupRetentionPeriod(dans la CLI :`--backup-retention-period`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Non applicable. La période de rétention des sauvegardes automatisées est gérée par le cluster de bases de données. Pour de plus amples informations, veuillez consulter the section called “Modifier DBCluster”.

  Par défaut : utilise le paramètre existant

- CACertificateIdentifier (dans la CLI : `--ca-certificate-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique le certificat qui doit être associé à l'instance.

- CloudwatchLogsExportConfiguration (dans la CLI : `--cloudwatch-logs-export-configuration`) : objet CloudwatchLogsExportConfiguration.

  Paramètre de configuration des types de journaux à activer pour l'exportation vers les CloudWatch journaux pour une instance de base de données ou un cluster de base de données spécifique.

- CopyTagsToSnapshot(dans la CLI :`--copy-tags-to-snapshot`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Valeur true pour copier toutes les balises de l'instance de base de données vers les instantanés de l'instance de base de données, et false dans le cas contraire. La valeur par défaut est false.

- DBInstanceClass (dans la CLI : `--db-instance-class`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nouvelle capacité de calcul et de mémoire de l'instance de base de données, par exemple `db.m4.large`. Toutes les classes d'instances de base de données ne sont pas disponibles dans toutes les régions Amazon.

La modification de la classe d'instance de base de données entraîne une interruption. La modification est appliquée pendant la fenêtre de maintenance suivante, sauf si `ApplyImmediately` a été spécifié `true` pour cette demande.

Par défaut : utilise le paramètre existant

- DBInstanceIdentifier (dans la CLI : `--db-instance-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de l'instance de base de données. Cette valeur est stockée sous la forme d'une chaîne en minuscules.

  Contraintes :

  - Doit correspondre à l'identifiant d'un existant DBInstance.
- DBParameterGroupName (dans la CLI : `--db-parameter-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de base de données à appliquer à l'instance de base de données. La modification de ce paramètre n'entraîne pas d'interruption. Le nom du groupe de paramètres lui-même est immédiatement changé, mais les modifications réelles des paramètres ne prennent effet qu'après le redémarrage de l'instance, sans basculement. L'instance de base de données NE sera PAS automatiquement redémarrée et les modifications du paramètre NE seront PAS appliquées pendant la fenêtre de maintenance suivante.

  Par défaut : utilise le paramètre existant

  Contraintes : le groupe de paramètres de base de données doit être dans la même famille de groupe de paramètres de base de données que cette instance de base de données.
- DBPortNumber(dans la CLI :`--db-port-number`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Numéro de port au niveau duquel la base de données accepte les connexions.

  La valeur du paramètre `DBPortNumber` ne doit correspondre à aucune des valeurs de port spécifiées pour les options du groupe d'options de l'instance de base de données.

  Votre base de données redémarre lorsque vous modifiez la valeur `DBPortNumber` quelle que soit la valeur du paramètre `ApplyImmediately`.

Par défaut : 8182

- DBSecurityGroups (dans la CLI : `--db-security-groups`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des Security Groups DB à autoriser sur cette instance de base de données. La modification de ce paramètre n'entraîne pas d'interruption et est appliquée de manière asynchrone dès que possible.

  Contraintes :

  - S'il est fourni, il doit correspondre aux DBSecurity groupes existants.
- DBSubnetGroupName (dans la CLI : `--db-subnet-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nouveau groupe de sous-réseaux de l'instance de base de données. Vous pouvez utiliser ce paramètre pour transférer votre instance de base de données vers un autre VPC.

  La modification du groupe de sous-réseaux entraîne une interruption. La modification est appliquée pendant la fenêtre de maintenance suivante, sauf si vous spécifiez `true` pour le paramètre `ApplyImmediately`.

  Contraintes : s'il est fourni, il doit correspondre au nom d'un DBSubnet groupe existant.

  Exemple : `mySubnetGroup`
- DeletionProtection(dans la CLI :`--deletion-protection`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Une valeur qui indique si la protection contre la suppression a été activée pour l'instance de base de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée. Par défaut, la protection contre la suppression est désactivée. Consultez Suppression d'une instance de base de données.
- Domain (dans la CLI : `--domain`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge.
- DomainIAMRoleName (dans la CLI : `--domain-iam-role-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge

- EnableIAMDatabaseAuthentication(dans la CLI :`--enable-iam-database-authentication`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Valeur true pour activer le mappage des comptes Amazon Identity and Access Management (IAM) avec les comptes de base de données ; sinon, valeur false.

  Vous pouvez activer l'authentification de base de données IAM pour les moteurs de base de données suivants

  Non applicable. Le mappage des comptes Amazon IAM avec des comptes de base de données est géré par le cluster de bases de données. Pour de plus amples informations, veuillez consulter the section called “Modifier DBCluster”.

  Par défaut: `false`

- EngineVersion (dans la CLI : `--engine-version`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Numéro de version du moteur de base de données vers lequel effectuer la mise à niveau. Actuellement, la définition de ce paramètre n'a aucun effet. Pour mettre à niveau votre moteur de base de données vers la version la plus récente, utilisez l'API the section called “ApplyPendingMaintenanceAction”.

- Iops(dans la CLI :`--iops`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nouvelle valeur d'IOPS provisionnés (opérations d'E/S par seconde) pour l'instance.

  La modification de ce paramètre n'entraîne pas d'interruption et la modification est appliquée pendant la fenêtre de maintenance suivante, sauf si le paramètre `ApplyImmediately` est défini sur `true` pour cette demande.

  Par défaut : utilise le paramètre existant

- MonitoringInterval(dans la CLI :`--monitoring-interval`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Intervalle, en secondes, entre les points lorsque des métriques de surveillance améliorée sont collectées pour l'instance de base de données. Pour désactiver la collecte des métriques de surveillance améliorée, spécifiez 0. La valeur par défaut est 0.

  Si `MonitoringRoleArn` est spécifié, vous devez également définir `MonitoringInterval` sur une valeur différente de 0.

Valeurs valides : `0, 1, 5, 10, 15, 30, 60`

- MonitoringRoleArn (dans la CLI : `--monitoring-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN du rôle IAM qui permet à Neptune d'envoyer des métriques de surveillance améliorées à Amazon CloudWatch Logs. Par exemple, `arn:aws:iam:123456789012:role/emaccess`.

  Si `MonitoringInterval` est défini sur une valeur différente de 0, vous devez fournir une valeur `MonitoringRoleArn`.

- MultiAZ(dans la CLI :`--multi-az`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si l'instance de base de données est un déploiement multi-AZ. La modification de ce paramètre n'entraîne pas d'interruption et la modification est appliquée pendant la fenêtre de maintenance suivante, sauf si le paramètre `ApplyImmediately` est défini sur `true` pour cette demande.

- NewDBInstanceIdentifier (dans la CLI : `--new-db-instance-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nouvel identifiant d'instance DB pour l'instance de base de données lors du renommage d'une instance de base de données. Lorsque vous modifiez l'identifiant d'instance DB, un redémarrage d'instance se produit immédiatement si vous définissez `Apply Immediately` sur true, ou se produira pendant la fenêtre de maintenance suivants si vous définissez `Apply Immediately` sur false. Cette valeur est stockée sous la forme d'une chaîne en minuscules.

  Contraintes :

  - Doit contenir entre 1 et 63 lettres, chiffres ou traits d'union.
  - Le premier caractère doit être une lettre.
  - Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs.

  Exemple : `mydbinstance`

- PreferredBackupWindow (dans la CLI : `--preferred-backup-window`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Intervalle de temps quotidien au cours duquel les sauvegardes automatiques sont créées si des sauvegardes automatiques sont activées.

Non applicable. L'intervalle de temps quotidien pour la création des sauvegardes automatisées est géré par le cluster de bases de données. Pour de plus amples informations, veuillez consulter the section called "Modifier DBCluster".

Contraintes :

- Doit être au format hh24:mi-hh24:mi
- Doit être dans le fuseau UTC (temps universel)
- Ne doit pas être en conflit avec la fenêtre de maintenance préférée
- Doit être de 30 minutes minimum.

- PreferredMaintenanceWindow (dans la CLI : `--preferred-maintenance-window`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Intervalle de temps hebdomadaire (au format UTC) pendant lequel se produit la maintenance du système qui peut entraîner une interruption. La modification de ce paramètre n'entraîne pas une interruption, sauf dans le cas suivant, et est appliquée de manière asynchrone dès que possible. Si des actions en attente entraînent un redémarrage et si la fenêtre de maintenance est modifiée pour inclure l'heure actuelle, la modification de ce paramètre entraînera un redémarrage de l'instance de base de données. Si vous définissez la fenêtre sur l'heure actuelle, vous devez prévoir 30 minutes minimum entre l'heure actuelle et la fin de la fenêtre afin que les modifications en attente soient appliquées.

  Par défaut : utilise le paramètre existant

  Format : ddd:hh24:mi-ddd:hh24:mi

  Jours valides : Mon | Tue | Wed | Thu | Fri | Sat | Sun

  Contraintes : doit être de 30 minutes minimum.

- PromotionTier(dans la CLI :`--promotion-tier`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Valeur qui spécifie l'ordre dans lequel un réplica en lecture est promu en instance principale après un échec de l'instance principale existante.

  Valeur par défaut : 1

  Valeurs valides : 0 - 15

- PubliclyAccessible(dans la CLI :`--publicly-accessible`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Cet indicateur ne doit plus être utilisé.
- StorageType (dans la CLI : `--storage-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non applicable. Dans Neptune, le type de stockage est géré au niveau du cluster de bases de données.
- TdeCredentialArn (dans la CLI : `--tde-credential-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN du magasin de clés auquel associer l'instance pour le chiffrement TDE.
- TdeCredentialPassword(dans la CLI :`--tde-credential-password`) — a SensitiveString, de type : `string` (une chaîne codée en UTF-8).

  Mot de passe de l'ARN donnée du magasin de clés pour accéder à l'appareil.
- VpcSecurityGroupIds (dans la CLI : `--vpc-security-group-ids`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des groupes de sécurité EC2 VPC à autoriser sur cette instance de base de données. Cette modification est appliquée de manière asynchrone dès que possible.

  Non applicable. La liste associée des groupes de sécurité EC2 VPC est gérée par le cluster de base de données. Pour de plus amples informations, veuillez consulter the section called “Modifier DBCluster”.

  Contraintes :
  - S'il est fourni, il doit correspondre à l'existant VpcSecurityGroupIds.

Réponse

Contient les détails d'une instance de base de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called “Décrivez DBInstances”.

- AutoMinorVersionUpgrade : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

Indique que des correctifs de versions mineures sont appliquées automatiquement.

- AvailabilityZone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom de la zone de disponibilité dans laquelle l'instance DB se trouve.

- BackupRetentionPeriod : entier de type : `integer` (entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.

- CACertificateIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du certificat CA de cette instance de base de données.

- CopyTagsToSnapshot : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si des balises sont copiées de l'instance de base de données vers des instantanés de l'instance de base de données.

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si l'instance de base de données est membre d'un cluster de bases de données, elle contient le nom du cluster de bases de données dont elle est membre.

- DBInstanceArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'instance de base de données.

- DBInstanceClass : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nom de la classe de capacité de calcul et de mémoire de l'instance de base de données.

- DBInstanceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de base de données fourni par l'utilisateur. Ce dernier est la clé unique qui identifie une instance de base de données.

- DbInstancePort : entier de type : `integer` (entier signé de 32 bits).

  Spécifie le port sur lequel l'instance de base de données écoute. Si l'instance de base de données fait partie d'un cluster de bases de données, le port peut être différent du port de cluster de bases de données.

- DBInstanceStatus : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique l'état actuel de cette base de données.

- DbiResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à chaque région Amazon pour l'instance de base de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS de l'instance de base de données est consultée.

- DBName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la base de données.

- DBParameterGroups : tableau d'objets DBParameterGroupStatus.

  Fournit la liste des groupes de paramètres de base de données appliqués à cette instance de base de données.

- DBSecurityGroups : tableau d'objets DBSecurityGroupMembership.

  Fournit la liste des éléments du Security Group DB contenant uniquement des sous-éléments `DBSecurityGroup.Name` et `DBSecurityGroup.Status`.

- DBSubnetGroup : objet DBSubnetGroupe.

  Spécifie des informations sur le groupe de sous-réseaux associé à l'instance de base de données, dont le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si la protection contre la suppression a été activée pour l'instance de base de données. L'instance ne peut pas être supprimée tant que la protection contre la suppression est activée. Consultez Suppression d'une instance de base de données.

- DomainMemberships : tableau d'objets DomainMembership.

  Non pris en charge

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux que cette instance de base de données est configurée pour exporter vers CloudWatch Logs.

- Endpoint : objet Point de terminaison.

  Spécifie le point de terminaison de la connexion.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

Fournit le nom du moteur de base de données à utiliser pour cette instance de base de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.

- EnhancedMonitoringResourceArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le nom de ressource Amazon (ARN) du flux de journal Amazon CloudWatch Logs qui reçoit les données de métriques de surveillance améliorée pour l'instance de base de données.

- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true si l'authentification Amazon Identity and Access Management (IAM) est activée, et false dans le cas contraire.

- InstanceCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Indique la date et l'heure de création de l'instance de base de données.

- Iops— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie la valeur des IOPS provisionnés (opérations d'E/S par seconde).

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge : Le chiffrement des instances de base de données est géré par le cluster de bases de données.

- LatestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée par point-in-time restauration.

- LicenseModel : chaîne de type : `string` (chaîne encodée en UTF-8).

  Informations sur le modèle de licence de cette instance de base de données.

- MonitoringInterval— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Intervalle, en secondes, entre les points lorsque des métriques de surveillance améliorée sont collectées pour l'instance de base de données.

- MonitoringRoleArn : chaîne de type : `string` (chaîne encodée en UTF-8).

L'ARN du rôle IAM qui permet à Neptune d'envoyer des métriques de surveillance améliorée à Amazon CloudWatch Logs.

- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si l'instance de base de données est un déploiement multi-AZ.

- PendingModifiedValues : objet PendingModifiedValues.

  Spécifie que les modifications apportées à l'instance de base de données sont en attente. Cet élément est inclus uniquement lorsque des modifications sont en attente. Des modifications spécifiques sont identifiées par sous-éléments.

- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.

- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.

- PromotionTier— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Valeur qui spécifie l'ordre dans lequel un réplica en lecture est promu en instance principale après un échec de l'instance principale existante.

- PubliclyAccessible : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Cet indicateur ne doit plus être utilisé.

- ReadReplicaDBClusterIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des clusters de bases de données qui sont des réplicas en lecture de cette instance de base de données.

- ReadReplicaDBInstanceIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à cette instance de base de données.

- ReadReplicaSourceDBInstanceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

Contient l'identifiant de l'instance de base de données source si cette dernière est un réplica en lecture.

- SecondaryAvailabilityZone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si cette valeur est présente, spécifie le nom de la zone de disponibilité secondaire d'une instance de base de données avec prise en charge multi-AZ.

- StatusInfos : tableau d'objets DBInstanceStatusInfo.

  Statut d'un réplica en lecture. Si l'instance n'est pas un réplica en lecture, ce champ est vide.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Non pris en charge : Le chiffrement des instances de base de données est géré par le cluster de bases de données.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le type de stockage associé à l'instance de base de données.

- TdeCredentialArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN du magasin de clés associé à l'instance pour le chiffrement TDE.

- Timezone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge.

- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit une liste des éléments du groupe de sécurité VPC auxquels appartient l'instance de base de données.

Erreurs

- Non valide DBInstance StateFault
- Non valide DBSecurity GroupStateFault
- DBInstanceAlreadyExistsFault
- DBInstanceNotFoundFault
- DBSecurityGroupNotFoundFault
- DBParameterGroupNotFoundFault

- Insuffisant DBInstance CapacityFault
- StorageQuotaExceededFault
- Non valide VPCNetwork StateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBUpgradeDependencyFailureFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- CertificateNotFoundFault
- DomainNotFoundFault

Redémarrer DBInstance (action)

Le nom de la AWS CLI pour cette API est :`reboot-db-instance`.

Vous pouvez avoir besoin de redémarrer votre instance de bases de données, en général pour des raisons de maintenance. Par exemple, si vous effectuez certaines modifications, ou si vous changez un groupe de paramètres de base de données associé à l'instance de bases de données, vous devez redémarrer l'instance pour que les modifications prennent effet.

Le redémarrage d'une instance de base de données entraîne celui du service du moteur de base de données. Le redémarrage d'une instance de bases de données entraîne une interruption momentanée, au cours de laquelle le statut de l'instance de bases de données est défini sur redémarrage.

Demande

- DBInstanceIdentifier (dans la CLI : `--db-instance-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de l'instance de base de données. Ce paramètre est stocké sous la forme d'une chaîne en lettres minuscules.

  Contraintes :

  - Doit correspondre à l'identifiant d'un existant DBInstance.

- ForceFailover(dans la CLI :`--force-failover`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Lorsque la valeur est `true`, le redémarrage se déroule via un basculement MultiAZ.

  Contrainte : vous ne pouvez pas spécifier `true` si l'instance n'est pas configurée pour MultiAZ.

Réponse

Contient les détails d'une instance de base de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called “Décrivez DBInstances”.

- AutoMinorVersionUpgrade : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique que des correctifs de versions mineures sont appliquées automatiquement.
- AvailabilityZone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom de la zone de disponibilité dans laquelle l'instance DB se trouve.
- BackupRetentionPeriod : entier de type : `integer` (entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.
- CACertificateIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du certificat CA de cette instance de base de données.
- CopyTagsToSnapshot : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si des balises sont copiées de l'instance de base de données vers des instantanés de l'instance de base de données.
- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si l'instance de base de données est membre d'un cluster de bases de données, elle contient le nom du cluster de bases de données dont elle est membre.
- DBInstanceArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'instance de base de données.
- DBInstanceClass : chaîne de type : `string` (chaîne encodée en UTF-8).

Contient le nom de la classe de capacité de calcul et de mémoire de l'instance de base de données.

- DBInstanceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de base de données fourni par l'utilisateur. Ce dernier est la clé unique qui identifie une instance de base de données.

- DbInstancePort : entier de type : `integer` (entier signé de 32 bits).

  Spécifie le port sur lequel l'instance de base de données écoute. Si l'instance de base de données fait partie d'un cluster de bases de données, le port peut être différent du port de cluster de bases de données.

- DBInstanceStatus : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique l'état actuel de cette base de données.

- DbiResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à chaque région Amazon pour l'instance de base de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS de l'instance de base de données est consultée.

- DBName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la base de données.

- DBParameterGroups : tableau d'objets DBParameterGroupStatus.

  Fournit la liste des groupes de paramètres de base de données appliqués à cette instance de base de données.

- DBSecurityGroups : tableau d'objets DBSecurityGroupMembership.

  Fournit la liste des éléments du Security Group DB contenant uniquement des sous-éléments `DBSecurityGroup.Name` et `DBSecurityGroup.Status`.

- DBSubnetGroup : objet DBSubnetGroupe.

  Spécifie des informations sur le groupe de sous-réseaux associé à l'instance de base de données, dont le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

Indique si la protection contre la suppression a été activée pour l'instance de base de données. L'instance ne peut pas être supprimée tant que la protection contre la suppression est activée. Consultez Suppression d'une instance de base de données.

- DomainMemberships : tableau d'objets DomainMembership.

  Non pris en charge

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux que cette instance de base de données est configurée pour exporter vers CloudWatch Logs.

- Endpoint : objet Point de terminaison.

  Spécifie le point de terminaison de la connexion.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour cette instance de base de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.

- EnhancedMonitoringResourceArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le nom de ressource Amazon (ARN) du flux de journal Amazon CloudWatch Logs qui reçoit les données de métriques de surveillance améliorée pour l'instance de base de données.

- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true si l'authentification Amazon Identity and Access Management (IAM) est activée, et false dans le cas contraire.

- InstanceCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Indique la date et l'heure de création de l'instance de base de données.

- Iops— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie la valeur des IOPS provisionnés (opérations d'E/S par seconde).

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

Non pris en charge : Le chiffrement des instances de base de données est géré par le cluster de bases de données.

- LatestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée par point-in-time restauration.

- LicenseModel : chaîne de type : `string` (chaîne encodée en UTF-8).

  Informations sur le modèle de licence de cette instance de base de données.

- MonitoringInterval— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Intervalle, en secondes, entre les points lorsque des métriques de surveillance améliorée sont collectées pour l'instance de base de données.

- MonitoringRoleArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN du rôle IAM qui permet à Neptune d'envoyer des métriques de surveillance améliorée à Amazon CloudWatch Logs.

- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si l'instance de base de données est un déploiement multi-AZ.

- PendingModifiedValues : objet PendingModifiedValues.

  Spécifie que les modifications apportées à l'instance de base de données sont en attente. Cet élément est inclus uniquement lorsque des modifications sont en attente. Des modifications spécifiques sont identifiées par sous-éléments.

- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.

- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.

- PromotionTier— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

Valeur qui spécifie l'ordre dans lequel un réplica en lecture est promu en instance principale après un échec de l'instance principale existante.

- PubliclyAccessible : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Cet indicateur ne doit plus être utilisé.

- ReadReplicaDBClusterIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des clusters de bases de données qui sont des réplicas en lecture de cette instance de base de données.

- ReadReplicaDBInstanceIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à cette instance de base de données.

- ReadReplicaSourceDBInstanceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient l'identifiant de l'instance de base de données source si cette dernière est un réplica en lecture.

- SecondaryAvailabilityZone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si cette valeur est présente, spécifie le nom de la zone de disponibilité secondaire d'une instance de base de données avec prise en charge multi-AZ.

- StatusInfos : tableau d'objets DBInstanceStatusInfo.

  Statut d'un réplica en lecture. Si l'instance n'est pas un réplica en lecture, ce champ est vide.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Non pris en charge : Le chiffrement des instances de base de données est géré par le cluster de bases de données.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le type de stockage associé à l'instance de base de données.

- TdeCredentialArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN du magasin de clés associé à l'instance pour le chiffrement TDE.

- Timezone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge.

- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit une liste des éléments du groupe de sécurité VPC auxquels appartient l'instance de base de données.

Erreurs

- Non valide DBInstance StateFault
- DBInstanceNotFoundFault

Décrire DBInstances (action)

Le nom de la AWS CLI pour cette API est :`describe-db-instances`.

Renvoie des informations sur les instances provisionnées et prend en charge la pagination.

> **Note**
>
> Cette opération peut également renvoyer des informations pour les instances Amazon RDS et les instances Amazon DocDB.

Demande

- DBInstanceIdentifier (dans la CLI : `--db-instance-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant d'instance fourni par l'utilisateur. Si ce paramètre est spécifié, seules les informations de l'instance de base de données spécifique sont renvoyées. Ce paramètre n'est pas sensible à la casse.

  Contraintes :

  - S'il est fourni, il doit correspondre à l'identifiant d'un existant DBInstance.
- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Filtre qui spécifie une ou plusieurs instances de bases de données à décrire.

  Filtres pris en charge :

- `db-cluster-id`- Accepte les identifiants de cluster de base de données et les Amazon Resource Names (ARNs) du cluster de bases de données. La liste des résultats inclura uniquement des informations sur les instances de base de données associées aux clusters de base de données identifiés par celles-ci ARNs.
- `engine` - Accepte un nom de moteur (tel que `neptune`) et restreint la liste de résultats aux instances de base de données créées par ce moteur.

Par exemple, pour invoquer cette API à partir de l'interface de ligne de commande Amazon et effectuer un filtrage afin que seules les instances de base de données Neptune soient renvoyées, vous pouvez utiliser la commande suivante :

Example

```
aws neptune describe-db-instances \
            --filters  Name=engine,Values=neptune
```

- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande `DescribeDBInstances` précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.
- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.

Réponse

- DBInstances : tableau d'objets DBInstance.

  Liste des instances the section called "DBInstance".
- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

Jeton de pagination facultatif fourni par une demande précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

Erreurs

- DBInstanceNotFoundFault

DescribeOrderableDBInstanceOptions (action)

Le nom de la AWS CLI pour cette API est :`describe-orderable-db-instance-options`.

Renvoie une liste des options d'instance de base de données à commander pour le moteur spécifié.

Demande

- DBInstanceClass (dans la CLI : `--db-instance-class`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Valeur de filtre de la classe d'instance de base de données. Spécifiez ce paramètre pour afficher uniquement les offres disponibles correspondant à la classe d'instance de base de données.
- Engine (dans la CLI : `--engine`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du moteur pour lequel récupérer les options d'instance de base de données.
- EngineVersion (dans la CLI : `--engine-version`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Valeur de filtre de la version de moteur. Spécifiez ce paramètre pour afficher uniquement les offres disponibles correspondant à la version de moteur spécifiée.
- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Ce paramètre n'est actuellement pas pris en charge.
- LicenseModel (dans la CLI : `--license-model`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Valeur de filtre du modèle de licence. Spécifiez ce paramètre pour afficher uniquement les offres disponibles correspondant au modèle de licence spécifié.

- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Un jeton de pagination facultatif fourni par une demande d' DescribeOrderableDBInstanceoptions précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.

- Vpc(dans la CLI :`--vpc`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Valeur de filtre VPC. Spécifiez ce paramètre pour afficher uniquement les offres VPC ou non VPC disponibles.

Réponse

- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Un jeton de pagination facultatif fourni par une précédente demande d'DBInstanceoptions commandables. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

- OrderableDBInstanceOptions : tableau d'objets Option commandable DBInstance.

  Structure the section called "Option commandable DBInstance" contenant des informations sur les options à commander pour l'instance de base de données.

DescribeValidDBInstanceModifications (action)

Le nom de la AWS CLI pour cette API est :`describe-valid-db-instance-modifications`.

Vous pouvez appeler the section called "DescribeValidDBInstanceModifications" pour connaître les modifications à apporter à votre instance de base de données. Vous pouvez utiliser ces informations lorsque vous appelez the section called "Modifier DBInstance".

Demande

- DBInstanceIdentifier (dans la CLI : `--db-instance-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant client ou ARN de votre instance de base de données.

Réponse

Informations sur les modifications valides que vous pouvez apporter à votre instance de base de données. Contient le résultat d'un appel réussi à l'action the section called "DescribeValidDBInstanceModifications". Vous pouvez utiliser ces informations lorsque vous appelez the section called "Modifier DBInstance".

- Storage : tableau d'objets ValidStorageOptions.

  Options de stockage valides pour votre instance de base de données.

Erreurs

- DBInstanceNotFoundFault
- Non valide DBInstance StateFault

Structures :

DBInstance (structure)

Contient les détails d'une instance de base de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called "Décrivez DBInstances".

Champs

- AutoMinorVersionUpgrade : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique que des correctifs de versions mineures sont appliquées automatiquement.
- AvailabilityZone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom de la zone de disponibilité dans laquelle l'instance DB se trouve.
- BackupRetentionPeriod : entier de type : `integer` (entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.
- CACertificateIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du certificat CA de cette instance de base de données.
- CopyTagsToSnapshot : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si des balises sont copiées de l'instance de base de données vers des instantanés de l'instance de base de données.
- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si l'instance de base de données est membre d'un cluster de bases de données, elle contient le nom du cluster de bases de données dont elle est membre.
- DBInstanceArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'instance de base de données.
- DBInstanceClass : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nom de la classe de capacité de calcul et de mémoire de l'instance de base de données.
- DBInstanceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de base de données fourni par l'utilisateur. Ce dernier est la clé unique qui identifie une instance de base de données.
- DbInstancePort : entier de type : `integer` (entier signé de 32 bits).

Spécifie le port sur lequel l'instance de base de données écoute. Si l'instance de base de données fait partie d'un cluster de bases de données, le port peut être différent du port de cluster de bases de données.

- DBInstanceStatus : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique l'état actuel de cette base de données.

- DbiResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à chaque région Amazon pour l'instance de base de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS de l'instance de base de données est consultée.

- DBName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la base de données.

- DBParameterGroups : tableau d'objets DBParameterGroupStatus.

  Fournit la liste des groupes de paramètres de base de données appliqués à cette instance de base de données.

- DBSecurityGroups : tableau d'objets DBSecurityGroupMembership.

  Fournit la liste des éléments du Security Group DB contenant uniquement des sous-éléments `DBSecurityGroup.Name` et `DBSecurityGroup.Status`.

- DBSubnetGroup : objet DBSubnetGroupe.

  Spécifie des informations sur le groupe de sous-réseaux associé à l'instance de base de données, dont le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— Il s'agit d'une BooleanOptional valeur booléenne `boolean` (vraie ou fausse) du type : (valeur booléenne).

  Indique si la protection contre la suppression a été activée pour l'instance de base de données. L'instance ne peut pas être supprimée tant que la protection contre la suppression est activée. Consultez Suppression d'une instance de base de données.

- DomainMemberships : tableau d'objets DomainMembership.

  Non pris en charge

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

Liste des types de journaux que cette instance de base de données est configurée pour exporter vers CloudWatch Logs.

- Endpoint : objet Point de terminaison.

  Spécifie le point de terminaison de la connexion.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour cette instance de base de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.

- EnhancedMonitoringResourceArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le nom de ressource Amazon (ARN) du flux de journal Amazon CloudWatch Logs qui reçoit les données de métriques de surveillance améliorée pour l'instance de base de données.

- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true si l'authentification Amazon Identity and Access Management (IAM) est activée, et false dans le cas contraire.

- InstanceCreateTime— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Indique la date et l'heure de création de l'instance de base de données.

- Iops— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie la valeur des IOPS provisionnés (opérations d'E/S par seconde).

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge : Le chiffrement des instances de base de données est géré par le cluster de bases de données.

- LatestRestorableTime— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée par point-in-time restauration.

- LicenseModel : chaîne de type : `string` (chaîne encodée en UTF-8).

Informations sur le modèle de licence de cette instance de base de données.

- MonitoringInterval— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Intervalle, en secondes, entre les points lorsque des métriques de surveillance améliorée sont collectées pour l'instance de base de données.

- MonitoringRoleArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN du rôle IAM qui permet à Neptune d'envoyer des métriques de surveillance améliorée à Amazon CloudWatch Logs.

- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si l'instance de base de données est un déploiement multi-AZ.

- PendingModifiedValues : objet PendingModifiedValues.

  Spécifie que les modifications apportées à l'instance de base de données sont en attente. Cet élément est inclus uniquement lorsque des modifications sont en attente. Des modifications spécifiques sont identifiées par sous-éléments.

- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.

- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.

- PromotionTier— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Valeur qui spécifie l'ordre dans lequel un réplica en lecture est promu en instance principale après un échec de l'instance principale existante.

- PubliclyAccessible : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Cet indicateur ne doit plus être utilisé.

- ReadReplicaDBClusterIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des clusters de bases de données qui sont des réplicas en lecture de cette instance de base de données.

- ReadReplicaDBInstanceIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à cette instance de base de données.

- ReadReplicaSourceDBInstanceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient l'identifiant de l'instance de base de données source si cette dernière est un réplica en lecture.

- SecondaryAvailabilityZone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si cette valeur est présente, spécifie le nom de la zone de disponibilité secondaire d'une instance de base de données avec prise en charge multi-AZ.

- StatusInfos : tableau d'objets DBInstanceStatusInfo.

  Statut d'un réplica en lecture. Si l'instance n'est pas un réplica en lecture, ce champ est vide.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Non pris en charge : Le chiffrement des instances de base de données est géré par le cluster de bases de données.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le type de stockage associé à l'instance de base de données.

- TdeCredentialArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN du magasin de clés associé à l'instance pour le chiffrement TDE.

- Timezone : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge.

- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit une liste des éléments du groupe de sécurité VPC auxquels appartient l'instance de base de données.

`DBInstance` est utilisé comme élément de réponse pour :

- Créez DBInstance
- Supprimer DBInstance

- Modifier DBInstance
- Redémarrer DBInstance

DBInstanceStatusInfo (structure)

Fournit une liste des informations sur le statut d'une instance de base de données.

Champs

- Message : chaîne de type : `string` (chaîne encodée en UTF-8).

  Détails de l'erreur en cas d'erreur de l'instance. Si l'instance n'est pas dans un état d'erreur, cette valeur est vide.
- Normal : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur booléenne true si l'instance fonctionne normalement, et false si l'instance est dans un état d'erreur.
- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Statut de l'instance de base de données. Pour une StatusType réplique en lecture, les valeurs peuvent être répliquées, erronées, arrêtées ou terminées.
- StatusType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Cette valeur est actuellement « read replication ».

DBInstanceOption commandable (structure)

Contient une liste des options disponibles pour une instance de base de données.

Ce type de données est utilisé comme élément de réponse dans l'action the section called "DescribeOrderableDBInstanceOptions".

Champs

- AvailabilityZones : tableau d'objets AvailabilityZone.

  Une liste des zones de disponibilité pour une instance de base de données.
- DBInstanceClass : chaîne de type : `string` (chaîne encodée en UTF-8).

Classe d'instance de base de données d'une instance de base de données.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de moteur d'une instance de base de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Version du moteur d'une instance de base de données.

- LicenseModel : chaîne de type : `string` (chaîne encodée en UTF-8).

  Modèle de licence d'une instance de base de données.

- MaxIopsPerDbInstance— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre total maximum des IOPS provisionnés pour une instance de base de données.

- MaxIopsPerGib— Il s'agit d'un DoubleOptional, de type : `double` (un nombre à virgule flottante IEEE 754 à double précision).

  Nombre total maximum des IOPS provisionnés par Gio pour une instance de base de données.

- MaxStorageSize— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Taille de stockage maximum d'une instance de base de données.

- MinIopsPerDbInstance— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre total minimum des IOPS provisionnés pour une instance de base de données.

- MinIopsPerGib— Il s'agit d'un DoubleOptional, de type : `double` (un nombre à virgule flottante IEEE 754 à double précision).

  Nombre total minimum des IOPS provisionnés par Gio pour une instance de base de données.

- MinStorageSize— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Taille de stockage minimum d'une instance de base de données.

- MultiAZCapable : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si une instance de base de données peut gérer le mode Multi-AZ.

- ReadReplicaCapable : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

Indique si une instance de base de données peut disposer d'un réplica en lecture.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non applicable. Dans Neptune, le type de stockage est géré au niveau du cluster de bases de données.

- SupportsEnhancedMonitoring : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si une instance de base de données prend en charge la surveillance améliorée à des intervalles de 1 à 60 secondes.

- SupportsGlobalDatabases : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur qui indique si vous pouvez utiliser les bases de données globales Neptune avec une combinaison spécifique d'autres attributs du moteur de base de données.

- SupportsIAMDatabaseAuthentication : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si une instance de base de données prend en charge l'authentification de base de données IAM.

- SupportsIops : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si une instance de base de données prend en charge les IOPS provisionnés.

- SupportsStorageEncryption : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si une instance de base de données prend en charge le stockage chiffré.

- Vpc : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si une instance de base de données se trouve dans un VPC.

PendingModifiedValues (structure)

Ce type de données est utilisé comme élément de réponse dans l'action the section called “Modifier DBInstance”.

Champs

- AllocatedStorage— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Contient la nouvelle taille `AllocatedStorage` pour l'instance de base de données qui sera appliquée ou est actuellement appliquée.
- BackupRetentionPeriod— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le nombre de jours en attente pour lesquels des sauvegardes automatiques sont conservées.
- CACertificateIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'identifiant du certificat CA pour l'instance de base de données.
- DBInstanceClass : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nouveau paramètre `DBInstanceClass` pour l'instance de base de données qui sera appliqué ou est actuellement appliqué.
- DBInstanceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nouveau paramètre `DBInstanceIdentifier` pour l'instance de base de données qui sera appliqué ou est actuellement appliqué.
- DBSubnetGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nouveau groupe de sous-réseaux de l'instance de base de données.
- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.
- Iops— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie la nouvelle valeur d'IOPS provisionnés pour l'instance de base de données qui sera appliquée ou est actuellement appliquée.
- MultiAZ— Il s'agit d'une BooleanOptional valeur booléenne `boolean` (vraie ou fausse) du type : (valeur booléenne).

  Indique que l'instance de base de données mono-AZ doit être remplacée par un déploiement multi-AZ.
- PendingCloudwatchLogsExports : objet PendingCloudwatchLogsExports.

Cette `PendingCloudwatchLogsExports` structure indique les modifications en attente concernant CloudWatch les journaux activés et ceux qui sont désactivés.

- Port— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le port en attente pour l'instance de base de données.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non applicable. Dans Neptune, le type de stockage est géré au niveau du cluster de bases de données.

ValidStorageOptions (structure)

Non applicable. Dans Neptune, le type de stockage est géré au niveau du cluster de bases de données.

Champs

- IopsToStorageRatio : tableau d'objets DoubleRange.

  Non applicable. Dans Neptune, le type de stockage est géré au niveau du cluster de bases de données.

- ProvisionedIops : tableau d'objets Range.

  Non applicable. Dans Neptune, le type de stockage est géré au niveau du cluster de bases de données.

- StorageSize : tableau d'objets Range.

  Non applicable. Dans Neptune, le type de stockage est géré au niveau du cluster de bases de données.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non applicable. Dans Neptune, le type de stockage est géré au niveau du cluster de bases de données.

Valide DBInstance ModificationsMessage (structure)

Informations sur les modifications valides que vous pouvez apporter à votre instance de base de données. Contient le résultat d'un appel réussi à l'action the section called "DescribeValidDBInstanceModifications". Vous pouvez utiliser ces informations lorsque vous appelez the section called "Modifier DBInstance".

Champs

- Storage : tableau d'objets ValidStorageOptions.

  Options de stockage valides pour votre instance de base de données.

`ValidDBInstanceModificationsMessage` est utilisé comme élément de réponse pour :

- DescribeValidDBInstanceModifications

API de paramètres Neptune

Actions :

- Copier DBParameter le groupe (action)
- Copier DBCluster ParameterGroup (action)
- Créer un DBParameter groupe (action)
- Créer DBCluster ParameterGroup (action)
- Supprimer le DBParameter groupe (action)
- Supprimer DBCluster ParameterGroup (action)
- Modifier DBParameter le groupe (action)
- Modifier DBCluster ParameterGroup (action)
- Réinitialiser DBParameter le groupe (action)
- Réinitialiser DBCluster ParameterGroup (action)
- Décrire DBParameters (action)
- Décrire DBParameter les groupes (action)
- Décrire DBCluster les paramètres (action)

- Décrire DBCluster ParameterGroups (action)
- DescribeEngineDefaultParameters (action)
- DescribeEngineDefaultClusterParameters (action)

Structures :

- Parameter (structure)
- DBParameterGroupe (structure)
- DBClusterParameterGroup (structure)
- DBParameterGroupStatus (structure)

Copier DBParameter le groupe (action)

Le nom de la AWS CLI pour cette API est :`copy-db-parameter-group`.

Copie le groupe de paramètres de base de données spécifié.

Demande

- SourceDBParameterGroupIdentifier (dans la CLI : `--source-db-parameter-group-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant ou ARN du groupe de paramètres de base de données source. Pour plus d'informations sur la création d'un ARN, consultez Création d'un Amazon Resource Name (ARN).

  Contraintes :

  - Doit spécifier un groupe de paramètres de base de données valide.
  - Doit spécifier un identifiant de groupe de paramètres de base de données valide, par exemple `my-db-param-group`, ou un ARN valide.
- Tags (dans la CLI : `--tags`) : tableau d'objets Tag.

  Balises à attribuer au groupe de paramètres de base de données copié.
- TargetDBParameterGroupDescription (dans la CLI : `--target-db-parameter-group-description`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Description du groupe de paramètres de base de données copié.

- TargetDBParameterGroupIdentifier (dans la CLI : `--target-db-parameter-group-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du groupe de paramètres de base de données copié.

  Contraintes :

  - Ne peut pas être null ou vide.
  - Doit contenir entre 1 et 255 lettres, chiffres ou traits d'union.
  - Le premier caractère doit être une lettre.
  - Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs.

  Exemple : `my-db-parameter-group`

Réponse

Contient les détails d'un groupe de paramètres de base de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called "Décrire DBParameter les groupes".

- DBParameterGroupArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du groupe de paramètres de base de données.
- DBParameterGroupFamily : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom de la famille de groupe de paramètres de base de données avec laquelle ce groupe de paramètres de base de données est compatible.
- DBParameterGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du groupe de paramètres de base de données.
- Description : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la description spécifiée par le client de ce groupe de paramètres de base de données.

Erreurs

- DBParameterGroupNotFoundFault
- DBParameterGroupAlreadyExistsFault

- DBParameterGroupQuotaExceededFault

Copier DBCluster ParameterGroup (action)

Le nom de la AWS CLI pour cette API est :`copy-db-cluster-parameter-group`.

Copie le groupe de paramètres de cluster de bases de données spécifié.

Demande

- SourceDBClusterParameterGroupIdentifier (dans la CLI : `--source-db-cluster-parameter-group-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant ou Amazon Resource Name (ARN) du groupe de paramètres de cluster de bases de données source. Pour plus d'informations sur la création d'un ARN, consultez Création d'un Amazon Resource Name (ARN).

  Contraintes :
  - Doit spécifier un groupe de paramètres de cluster de bases de données valide.
  - Si le groupe de paramètres de cluster de bases de données source se trouve dans la même région Amazon que la copie, spécifiez un identifiant de groupe de paramètres de base de données valide, par exemple `my-db-cluster-param-group`, ou un ARN valide.
  - Si le groupe de paramètres de base de données source se trouve dans une autre région Amazon que la copie, spécifiez un ARN de groupe de paramètres de cluster de bases de données valide, par exemple `arn:aws:rds:us-east-1:123456789012:cluster-pg:custom-cluster-group1`.
- Tags (dans la CLI : `--tags`) : tableau d'objets Tag.

  Balises à attribuer au groupe de paramètres de cluster de bases de données copié.
- TargetDBClusterParameterGroupDescription (dans la CLI : `--target-db-cluster-parameter-group-description`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Description du groupe de paramètres de cluster de bases de données copié.
- TargetDBClusterParameterGroupIdentifier (dans la CLI : `--target-db-cluster-parameter-group-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du groupe de paramètres de cluster de bases de données copié.

Contraintes :

- Ne peut pas être null ou vide
- Doit contenir entre 1 et 255 lettres, chiffres ou traits d'union
- Le premier caractère doit être une lettre
- Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs

Exemple : `my-cluster-param-group1`

Réponse

Contient les détails d'un groupe de paramètres de cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called "Décrivez DBCluster ParameterGroups".

- DBClusterParameterGroupArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du groupe de paramètres de cluster de bases de données.
- DBClusterParameterGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du groupe de paramètres du cluster de base de données.
- DBParameterGroupFamily : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom de la famille de groupe de paramètres de base de données avec laquelle ce groupe de paramètres de cluster de bases de données est compatible.
- Description : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la description spécifiée par le client de ce groupe de paramètres du cluster de bases de données.

Erreurs

- DBParameterGroupNotFoundFault
- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

Créer un DBParameter groupe (action)

Le nom de la AWS CLI pour cette API est :`create-db-parameter-group`.

Crée un groupe de paramètres de base de données.

Un groupe de paramètres de base de données est initialement créé avec les paramètres par défaut pour le moteur de base de données utilisé par l'instance de base de données. Pour fournir des valeurs personnalisées pour l'un des paramètres, vous devez modifier le groupe après l'avoir créé à l'aide de l'option Modifier le DBParameter groupe. Une fois que vous avez créé un groupe de paramètres de base de données, vous devez l'associer à votre instance de base de données à l'aide de Modify DBInstance. Lorsque vous associez un nouveau groupe de paramètres de base de données à une instance de base de données en cours d'exécution, vous devez redémarrer l'instance de base de données sans basculement, pour que le nouveau groupe de paramètres de base de données et les paramètres associés soient pris en compte.

> **Important**
>
> Après la création d'un groupe de paramètres DB, veuillez patienter au moins 5 minutes avant de créer votre première instance de base de données qui utilise ce groupe comme groupe de paramètres par défaut. Cela permet à Amazon Neptune de terminer complètement l'action de création avant que le groupe de paramètres soit utilisé par défaut pour une nouvelle instance de base de données. Cela est spécialement important pour des paramètres qui sont essentiels lors de la création de la base de données par défaut pour une instance de base de données, tels que le jeu de caractères pour la base de données par défaut définie par le paramètre `character_set_database`. Vous pouvez utiliser l'option Groupes de paramètres de la console Amazon Neptune ou la DBParameters commande Describe pour vérifier que votre groupe de paramètres de base de données a été créé ou modifié.

Demande

- DBParameterGroupFamily (dans la CLI : `--db-parameter-group-family`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la famille de groupe de paramètres de base de données. Un groupe de paramètres de base de données peut être associé à une seule famille de groupe de paramètres de base de données, et peut être uniquement appliqué à une instance de base de données exécutant un

moteur de base de données et une version de moteur compatibles avec cette famille de groupe de paramètres de base de données.

- DBParameterGroupName (dans la CLI : `--db-parameter-group-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de base de données.

  Contraintes :

  - Doit comporter entre 1 et 255 lettres, chiffres ou traits d'union.
  - Le premier caractère doit être une lettre
  - Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs

  > **Note**
  >
  > Cette valeur est stockée sous la forme d'une chaîne en minuscules.

- Description (dans la CLI : `--description`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Description du groupe de paramètres de base de données.

- Tags (dans la CLI : `--tags`) : tableau d'objets Tag.

  Balises à attribuer au nouveau groupe de paramètres de base de données.

Réponse

Contient les détails d'un groupe de paramètres de base de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called “Décrire DBParameter les groupes”.

- DBParameterGroupArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du groupe de paramètres de base de données.

- DBParameterGroupFamily : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom de la famille de groupe de paramètres de base de données avec laquelle ce groupe de paramètres de base de données est compatible.

- DBParameterGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du groupe de paramètres de base de données.
- Description : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la description spécifiée par le client de ce groupe de paramètres de base de données.

Erreurs

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

Créer DBCluster ParameterGroup (action)

Le nom de la AWS CLI pour cette API est :`create-db-cluster-parameter-group`.

Crée un groupe de paramètres de cluster de bases de données.

Les paramètres d'un groupe de paramètres de cluster de bases de données s'appliquent à toutes les instances d'un cluster de bases de données.

Un groupe de paramètres de cluster de bases de données est initialement créé avec les paramètres par défaut pour le moteur de base de données utilisé par les instances de base de données du cluster de bases de données. Pour fournir des valeurs personnalisées aux paramètres, vous devez modifier le groupe après sa création de l'aide de the section called “Modifier DBCluster ParameterGroup”. Après avoir créé un groupe de paramètres de cluster de bases de données, vous devez l'associer à votre cluster de bases de données à l'aide de the section called “Modifier DBCluster”. Lorsque vous associez un nouveau groupe de paramètres de cluster de bases de données à un cluster de bases de données en cours d'exécution, vous devez redémarrer les instances de base de données sans basculement, pour que le nouveau groupe de paramètres de cluster de bases de données et les paramètres associés soient pris en compte.

> **Important**
>
> Après avoir créé un groupe de paramètres de cluster de bases de données, vous devez patienter au moins 5 minutes avant de créer votre premier cluster de bases de données qui utilise ce groupe de paramètres de cluster de bases de données comme groupe de paramètres par défaut. Cela permet à Amazon Neptune de terminer complètement l'action de

création avant que le groupe de paramètres de cluster de bases de données soit utilisé par défaut pour un nouveau cluster de bases de données. Ce point est particulièrement important pour les paramètres essentiels lors de la création de la base de données par défaut pour un cluster de bases de données, tels que le jeu de caractères pour la base de données par défaut définie par le paramètre `character_set_database`. Vous pouvez utiliser l'option Groupe de paramètres de la console Amazon Neptune ou de la commande the section called "Décrire DBCluster les paramètres" pour vérifier que votre groupe de paramètres de cluster de bases de données a été créé ou modifié.

Demande

- DBClusterParameterGroupName (dans la CLI : `--db-cluster-parameter-group-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de cluster de bases de données.

  Contraintes :

  - Doit correspondre au nom d'un existant DBClusterParameterGroup.

  > **Note**
  >
  > Cette valeur est stockée sous la forme d'une chaîne en minuscules.

- DBParameterGroupFamily (dans la CLI : `--db-parameter-group-family`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la famille de groupe de paramètres de cluster de bases de données. Un groupe de paramètres de cluster de bases de données peut être associé à une seule famille de groupe de paramètres de cluster de bases de données, et peut être uniquement appliqué à un cluster de bases de données exécutant un moteur de base de données et une version de moteur compatibles avec cette famille de groupe de paramètres de cluster de bases de données.

- Description (dans la CLI : `--description`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Description du groupe de paramètres de cluster de bases de données.

- Tags (dans la CLI : `--tags`) : tableau d'objets Tag.

Balises à attribuer au nouveau groupe de paramètres de cluster de bases de données.

Réponse

Contient les détails d'un groupe de paramètres de cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called "Décrivez DBCluster ParameterGroups".

- DBClusterParameterGroupArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du groupe de paramètres de cluster de bases de données.
- DBClusterParameterGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du groupe de paramètres du cluster de base de données.
- DBParameterGroupFamily : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom de la famille de groupe de paramètres de base de données avec laquelle ce groupe de paramètres de cluster de bases de données est compatible.
- Description : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la description spécifiée par le client de ce groupe de paramètres du cluster de bases de données.

Erreurs

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

Supprimer le DBParameter groupe (action)

Le nom de la AWS CLI pour cette API est :`delete-db-parameter-group`.

Supprime un DBParameter groupe spécifié. Le DBParameter groupe à supprimer ne peut être associé à aucune instance de base de données.

Demande

- DBParameterGroupName (dans la CLI : `--db-parameter-group-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de base de données.

  Contraintes :

  - Doit être le nom d'un groupe de paramètres de base de données existant.
  - Vous ne pouvez pas supprimer un groupe de paramètres de base de données par défaut
  - Ne peut être associé à aucune instance de base de données

Réponse

- Paramètres d'absence de réponse.

Erreurs

- Non valide DBParameter GroupStateFault
- DBParameterGroupNotFoundFault

Supprimer DBCluster ParameterGroup (action)

Le nom de la AWS CLI pour cette API est :`delete-db-cluster-parameter-group`.

Supprime un groupe de paramètres de cluster de bases de données spécifié. Le groupe de paramètres de cluster de bases de données à supprimer ne peut être associé à aucun cluster de bases de données.

Demande

- DBClusterParameterGroupName (dans la CLI : `--db-cluster-parameter-group-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de cluster de bases de données.

  Contraintes :

  - Doit être le nom d'un groupe de paramètres de cluster de bases de données existant.
  - Vous ne pouvez pas supprimer un groupe de paramètres de cluster de bases de données par défaut.

- Ne peut être associé à aucun cluster de bases de données.

Réponse

- Paramètres d'absence de réponse.

Erreurs

- Non valide DBParameter GroupStateFault
- DBParameterGroupNotFoundFault

Modifier DBParameter le groupe (action)

Le nom de la AWS CLI pour cette API est :`modify-db-parameter-group`.

Modifie les paramètres d'un groupe de paramètres de base de données. Pour modifier plusieurs paramètres, soumettez une liste des éléments suivants : `ParameterName`, `ParameterValue` et `ApplyMethod`. 20 paramètres maximum peuvent être modifiés dans une même demande.

> **Note**
>
> Les modifications apportées aux paramètres dynamiques sont appliquées immédiatement. Avant leur entrée en vigueur, les modifications apportées aux paramètres statiques exigent un redémarrage sans basculement vers l'instance de base de données associée au groupe de paramètres.

> **Important**
>
> Après la modification d'un groupe de paramètres DB, veuillez patienter au moins 5 minutes avant de créer votre première instance DB utilisant ce groupe comme groupe de paramètres par défaut. Cela permet à Amazon Neptune de terminer complètement l'action de modification avant que le groupe de paramètres soit utilisé par défaut pour une nouvelle instance de base de données. Cela est spécialement important pour des paramètres qui sont essentiels lors de la création de la base de données par défaut pour une instance de base de données, tels que le jeu de caractères pour la base de données par défaut définie par le paramètre `character_set_database`. Vous pouvez utiliser l'option Groupes de

paramètres de la console Amazon Neptune ou la DBParameters commande Describe pour vérifier que votre groupe de paramètres de base de données a été créé ou modifié.

Demande

- DBParameterGroupName (dans la CLI : `--db-parameter-group-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de base de données.

  Contraintes :

  - S'il est fourni, il doit correspondre au nom d'un DBParameter groupe existant.
- Parameters (dans la CLI : `--parameters`) : obligatoire : tableau d'objets Paramètre.

  Tableau des noms de paramètres, des valeurs et méthode d'application pour la mise à jour des paramètres. Au moins un nom de paramètre, une valeur et la méthode d'application doivent être fournis ; les arguments suivants sont facultatifs. 20 paramètres maximum peuvent être modifiés dans une même demande.

  Valeur valides (pour la méthode d'application) : `immediate | pending-reboot`

  > **Note**
  >
  > Vous pouvez utiliser la valeur immédiate uniquement avec des paramètres dynamiques. Vous pouvez utiliser la valeur de redémarrage en attente pour les paramètres statiques et dynamiques, et les modifications sont appliquées lorsque vous redémarrez l'instance de base de données sans basculement.

Réponse

- DBParameterGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du groupe de paramètres de base de données.

Erreurs

- DBParameterGroupNotFoundFault

- Non valide DBParameter GroupStateFault

Modifier DBCluster ParameterGroup (action)

Le nom de la AWS CLI pour cette API est :`modify-db-cluster-parameter-group`.

Modifie les paramètres d'un groupe de paramètres de cluster de bases de données. Pour modifier plusieurs paramètres, soumettez une liste des éléments suivants : `ParameterName`, `ParameterValue` et `ApplyMethod`. 20 paramètres maximum peuvent être modifiés dans une même demande.

> **Note**
>
> Les modifications apportées aux paramètres dynamiques sont appliquées immédiatement. Avant leur entrée en vigueur, les modifications apportées aux paramètres statiques exigent un redémarrage sans basculement vers le cluster de bases de données associée au groupe de paramètres.

> **Important**
>
> Après avoir créé un groupe de paramètres de cluster de bases de données, vous devez patienter au moins 5 minutes avant de créer votre premier cluster de bases de données qui utilise ce groupe de paramètres de cluster de bases de données comme groupe de paramètres par défaut. Cela permet à Amazon Neptune de terminer complètement l'action de création avant que le groupe de paramètres soit utilisé par défaut pour un nouveau cluster de bases de données. Ce point est particulièrement important pour les paramètres essentiels lors de la création de la base de données par défaut pour un cluster de bases de données, tels que le jeu de caractères pour la base de données par défaut définie par le paramètre `character_set_database`. Vous pouvez utiliser l'option Groupe de paramètres de la console Amazon Neptune ou de la commande the section called “Décrire DBCluster les paramètres” pour vérifier que votre groupe de paramètres de cluster de bases de données a été créé ou modifié.

Demande

- DBClusterParameterGroupName (dans la CLI : `--db-cluster-parameter-group-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de cluster de bases de données à modifier.
- Parameters (dans la CLI : `--parameters`) : obligatoire : tableau d'objets Paramètre.

  Liste des paramètres du groupe de paramètres de cluster de bases de données à modifier.

Réponse

- DBClusterParameterGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de cluster de bases de données.

  Contraintes :
  - Doit comporter entre 1 et 255 lettres ou chiffres.
  - Le premier caractère doit être une lettre
  - Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs

  > **Note**
  >
  > Cette valeur est stockée sous la forme d'une chaîne en minuscules.

Erreurs

- DBParameterGroupNotFoundFault
- Non valide DBParameter GroupStateFault

Réinitialiser DBParameter le groupe (action)

Le nom de la AWS CLI pour cette API est :`reset-db-parameter-group`.

Modifie les paramètres d'un groupe de paramètres de base de données à la valeur engine/system par défaut. Pour réinitialiser des paramètres spécifiques, fournissez une liste des éléments suivants : `ParameterName` et `ApplyMethod`. Pour réinitialiser entièrement le groupe de paramètres de base de données, spécifiez le nom `DBParameterGroup` et les paramètres `ResetAllParameters`. Lors de la réinitialisation du groupe entier, les paramètres dynamiques sont mis à jour immédiatement et

les paramètres statiques sont définis sur `pending-reboot` pour entrer en vigueur lors du prochain redémarrage de l'instance de base de données ou de la demande `RebootDBInstance`.

Demande

- DBParameterGroupName (dans la CLI : `--db-parameter-group-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de base de données.

  Contraintes :

  - Doit correspondre au nom d'un DBParameter groupe existant.
- Parameters (dans la CLI : `--parameters`) : tableau d'objets Paramètre.

  Pour réinitialiser entièrement le groupe de paramètres de base de données, spécifiez le nom `DBParameterGroup` et les paramètres `ResetAllParameters`. Pour réinitialiser des paramètres spécifiques, fournissez une liste des éléments suivants : `ParameterName` et `ApplyMethod`. 20 paramètres maximum peuvent être modifiés dans une même demande.

  Valeurs valides (pour méthode d'application) : `pending-reboot`
- ResetAllParameters (dans la CLI : `--reset-all-parameters`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Spécifie s'il faut réinitialiser tous les paramètres du groupe de paramètres de base de données à leurs valeurs par défaut (`true`) ou pas (`false`).

  Par défaut : `true`

Réponse

- DBParameterGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du groupe de paramètres de base de données.

Erreurs

- Non valide DBParameter GroupStateFault
- DBParameterGroupNotFoundFault

Réinitialiser DBCluster ParameterGroup (action)

Le nom de la AWS CLI pour cette API est :`reset-db-cluster-parameter-group`.

Remplace les paramètres d'un groupe de paramètres de cluster de bases de données par les valeurs par défaut. Pour réinitialiser des paramètres spécifiques, envoyez une liste des éléments suivants : `ParameterName` et `ApplyMethod`. Pour réinitialiser entièrement le groupe de paramètres de cluster de bases de données, spécifiez les paramètres `DBClusterParameterGroupName` et `ResetAllParameters`.

Lors de la réinitialisation du groupe entier, les paramètres dynamiques sont mis à jour immédiatement et les paramètres statiques sont définis sur `pending-reboot` pour entrer en vigueur lors du prochain redémarrage de l'instance de base de données ou de la demande the section called "Redémarrer DBInstance". Vous devez appeler the section called "Redémarrer DBInstance" pour chaque instance de base de données de votre cluster de bases de données à laquelle vous souhaitez que la mise à jour des paramètres statiques soit appliquée.

Demande

- DBClusterParameterGroupName (dans la CLI : `--db-cluster-parameter-group-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de cluster de bases de données à réinitialiser.
- Parameters (dans la CLI : `--parameters`) : tableau d'objets Paramètre.

  Une liste des noms de paramètre du groupe de paramètres de cluster de bases de données à réinitialiser aux valeurs par défaut. Vous ne pouvez pas utiliser ce paramètre si le paramètre `ResetAllParameters` est défini sur `true`.
- ResetAllParameters (dans la CLI : `--reset-all-parameters`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur `true` pour réinitialiser tous les paramètres dans le groupe de paramètres de cluster de bases de données à leurs valeurs par défaut, et `false` dans le cas contraire. Vous ne pouvez pas utiliser ce paramètre si une liste des noms des paramètres spécifiés existe pour le paramètre `Parameters`.

Réponse

- DBClusterParameterGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

Nom du groupe de paramètres de cluster de bases de données.

Contraintes :

- Doit comporter entre 1 et 255 lettres ou chiffres.
- Le premier caractère doit être une lettre
- Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs

> **Note**
>
> Cette valeur est stockée sous la forme d'une chaîne en minuscules.

Erreurs

- Non valide DBParameter GroupStateFault
- DBParameterGroupNotFoundFault

Décrire DBParameters (action)

Le nom de la AWS CLI pour cette API est :`describe-db-parameters`.

Renvoie la liste détaillée des paramètres d'un groupe de paramètres de base de données spécifique.

Demande

- DBParameterGroupName (dans la CLI : `--db-parameter-group-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom d'un groupe de paramètres de base de données spécifique pour lequel renvoyer les détails.

  Contraintes :

  - S'il est fourni, il doit correspondre au nom d'un DBParameter groupe existant.
- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Ce paramètre n'est actuellement pas pris en charge.
- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

Jeton de pagination facultatif fourni par une demande `DescribeDBParameters` précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.

- Source (dans la CLI : `--source`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Types de paramètre à renvoyer.

  Par défaut : tous les types de paramètre sont renvoyés.

  Valeurs valides : `user | system | engine-default`

Réponse

- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

- Parameters : tableau d'objets Paramètre.

  Liste des valeurs the section called “Paramètre”.

Erreurs

- DBParameterGroupNotFoundFault

Décrire DBParameter les groupes (action)

Le nom de la AWS CLI pour cette API est :`describe-db-parameter-groups`.

Renvoie une liste des descriptions de `DBParameterGroup`. Si le paramètre `DBParameterGroupName` est spécifié, la liste contiendra uniquement la description du groupe de paramètres de base de données spécifié.

Demande

- DBParameterGroupName (dans la CLI : `--db-parameter-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom d'un groupe de paramètres de base de données spécifique pour lequel renvoyer les détails.

  Contraintes :

  - S'il est fourni, il doit correspondre au nom d'un existant DBClusterParameterGroup.
- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Ce paramètre n'est actuellement pas pris en charge.
- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande `DescribeDBParameterGroups` précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.
- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.

Réponse

- DBParameterGroups : tableau d'objets DBParameterGroupe.

Liste des instances the section called "DBParameterGroupe".

- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

Erreurs

- DBParameterGroupNotFoundFault

Décrire DBCluster les paramètres (action)

Le nom de la AWS CLI pour cette API est :`describe-db-cluster-parameters`.

Renvoie la liste détaillée des paramètres d'un groupe de paramètres de cluster de bases de données spécifique.

Demande

- DBClusterParameterGroupName (dans la CLI : `--db-cluster-parameter-group-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom d'un groupe de paramètres de cluster de bases de données spécifique pour lequel renvoyer les détails de paramètre.

  Contraintes :

  - S'il est fourni, il doit correspondre au nom d'un existant DBClusterParameterGroup.
- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Ce paramètre n'est actuellement pas pris en charge.
- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande `DescribeDBClusterParameters` précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.
- Source (dans la CLI : `--source`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Valeur indiquée pour renvoyer uniquement les paramètres d'une source spécifique. Les sources du paramètre peuvent être `engine`, `service` ou `customer`.

Réponse

- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Un jeton de pagination facultatif fourni par une précédente demande de description DBCluster des paramètres. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.
- Parameters : tableau d'objets Paramètre.

  Fournit une liste des paramètres du groupe de paramètres de cluster de bases de données.

Erreurs

- DBParameterGroupNotFoundFault

Décrire DBCluster ParameterGroups (action)

Le nom de la AWS CLI pour cette API est :`describe-db-cluster-parameter-groups`.

Renvoie une liste des descriptions de `DBClusterParameterGroup`. Si le paramètre `DBClusterParameterGroupName` est spécifié, la liste contiendra uniquement la description du groupe de paramètres de cluster de bases de données spécifié.

Demande

- DBClusterParameterGroupName (dans la CLI : `--db-cluster-parameter-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom d'un groupe de paramètres de cluster de bases de données spécifique pour lequel renvoyer les détails.

  Contraintes :

  - S'il est fourni, il doit correspondre au nom d'un existant DBClusterParameterGroup.
- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Ce paramètre n'est actuellement pas pris en charge.
- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande `DescribeDBClusterParameterGroups` précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.
- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.

Réponse

- DBClusterParameterGroups : tableau d'objets DBClusterParameterGroup.

  Liste des groupes de paramètres de cluster de bases de données.
- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande `DescribeDBClusterParameterGroups` précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

Erreurs

- DBParameterGroupNotFoundFault

DescribeEngineDefaultParameters (action)

Le nom de la AWS CLI pour cette API est :`describe-engine-default-parameters`.

Renvoie les informations sur les paramètres de moteur et système par défaut du moteur de base de données spécifié.

Demande

- DBParameterGroupFamily (dans la CLI : `--db-parameter-group-family`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la famille de groupe de paramètres de cluster de bases de données.
- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Cette option n'est pas prise en charge actuellement.
- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande `DescribeEngineDefaultParameters` précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.
- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.

Réponse

Contient le résultat d'une invocation réussie de l'action the section called "DescribeEngineDefaultParameters".

- DBParameterGroupFamily : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom de la famille de groupe de paramètres de base de données à laquelle s'appliquent les paramètres par défaut du moteur.
- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Un jeton de pagination facultatif fourni par une EngineDefaults demande précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.
- Parameters : tableau d'objets Paramètre.

  Contient une liste des paramètres par défaut du moteur.

DescribeEngineDefaultClusterParameters (action)

Le nom de la AWS CLI pour cette API est :`describe-engine-default-cluster-parameters`.

Renvoie les informations sur les paramètres de moteur et de système par défaut du moteur de base de données du cluster.

Demande

- DBParameterGroupFamily (dans la CLI : `--db-parameter-group-family`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la famille de groupe de paramètres de cluster de bases de données pour laquelle renvoyer les informations sur les paramètres de moteur.
- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Ce paramètre n'est actuellement pas pris en charge.
- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande `DescribeEngineDefaultClusterParameters` précédente. Si ce paramètre est spécifié, la

réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.

Réponse

Contient le résultat d'une invocation réussie de l'action the section called "DescribeEngineDefaultParameters".

- DBParameterGroupFamily : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom de la famille de groupe de paramètres de base de données à laquelle s'appliquent les paramètres par défaut du moteur.
- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Un jeton de pagination facultatif fourni par une EngineDefaults demande précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.
- Parameters : tableau d'objets Paramètre.

  Contient une liste des paramètres par défaut du moteur.

Structures :

Parameter (structure)

Spécifie un paramètre.

Champs

- AllowedValues : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de valeurs valide pour le paramètre.

- ApplyMethod— Il s'agit d'un ApplyMethod, de type : `string` (une chaîne codée en UTF-8).

  Indique quand appliquer les mises à jour de paramètres.

- ApplyType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le type de paramètres propres au moteur.

- DataType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le type de données valide pour le paramètre.

- Description : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit une description du paramètre.

- IsModifiable : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le paramètre peut être (`true`) ou non (`false`) modifié. Certains paramètres ont des implications en terme de sécurité ou de fonctionnement qui les empêchent d'être modifiés.

- MinimumEngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Première version de moteur à laquelle le paramètre peut s'appliquer.

- ParameterName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du paramètre.

- ParameterValue : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la valeur du paramètre.

- Source : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la source de la valeur du paramètre.

DBParameterGroupe (structure)

Contient les détails d'un groupe de paramètres de base de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called “Décrire DBParameter les groupes”.

Champs

- DBParameterGroupArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du groupe de paramètres de base de données.
- DBParameterGroupFamily : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom de la famille de groupe de paramètres de base de données avec laquelle ce groupe de paramètres de base de données est compatible.
- DBParameterGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du groupe de paramètres de base de données.
- Description : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la description spécifiée par le client de ce groupe de paramètres de base de données.

`DBParameterGroup` est utilisé comme élément de réponse pour :

- DBParameterGroupe de copies
- Créer un DBParameter groupe

DBClusterParameterGroup (structure)

Contient les détails d'un groupe de paramètres de cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called “Décrivez DBCluster ParameterGroups”.

Champs

- DBClusterParameterGroupArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du groupe de paramètres de cluster de bases de données.
- DBClusterParameterGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du groupe de paramètres du cluster de base de données.

- DBParameterGroupFamily : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom de la famille de groupe de paramètres de base de données avec laquelle ce groupe de paramètres de cluster de bases de données est compatible.
- Description : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la description spécifiée par le client de ce groupe de paramètres du cluster de bases de données.

`DBClusterParameterGroup` est utilisé comme élément de réponse pour :

- Copier DBCluster ParameterGroup
- Créez DBCluster ParameterGroup

DBParameterGroupStatus (structure)

Statut du groupe de paramètres de base de données.

Ce type de données est utilisé comme élément de réponse dans les actions suivantes :

- the section called "Créez DBInstance"
- the section called "Supprimer DBInstance"
- the section called "Modifier DBInstance"
- the section called "Redémarrer DBInstance"

Champs

- DBParameterGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de base de données.
- ParameterApplyStatus : chaîne de type : `string` (chaîne encodée en UTF-8).

  Statut des mises à jour de paramètres.

API de sous-réseau Neptune

Actions :

- Créer un DBSubnet groupe (action)
- Supprimer le DBSubnet groupe (action)
- Modifier DBSubnet le groupe (action)
- Décrire DBSubnet les groupes (action)

Structures :

- Subnet (structure)
- DBSubnetGroupe (structure)

Créer un DBSubnet groupe (action)

Le nom de la AWS CLI pour cette API est :`create-db-subnet-group`.

Crée un groupe de sous-réseaux de base de données. Les groupes de sous-réseaux de base de données doivent contenir au moins un sous-réseau sur au moins deux AZs dans la région Amazon.

Demande

- DBSubnetGroupDescription (dans la CLI : `--db-subnet-group-description`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Description du groupe de sous-réseaux de base de données.
- DBSubnetGroupName (dans la CLI : `--db-subnet-group-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de sous-réseaux de base de données. Cette valeur est stockée sous la forme d'une chaîne en minuscules.

  Contraintes : doit comporter au maximum 255 lettres, chiffres, points, traits de soulignement, espaces ou tirets. Impossible de conserver le nom par défaut.

  Exemple : `mySubnetgroup`
- SubnetIds (dans la CLI : `--subnet-ids`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  EC2 Sous-réseau du groupe IDs de sous-réseaux de base de données.

- Tags (dans la CLI : `--tags`) : tableau d'objets Tag.

  Balises à attribuer au nouveau groupe de sous-réseaux de base de données.

Réponse

Contient les détails d'un groupe de sous-réseaux de base de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called “Décrire DBSubnet les groupes”.

- DBSubnetGroupArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du groupe de sous-réseaux de base de données.
- DBSubnetGroupDescription : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la description du groupe de sous-réseaux de base de données.
- DBSubnetGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le nom du groupe de sous-réseaux DB.
- SubnetGroupStatus : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le statut du groupe de sous-réseaux de base de données.
- Subnets : tableau d'objets Sous-réseau.

  Contient une liste des éléments the section called “Sous-réseau”.
- VpcId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom VpcId du groupe de sous-réseaux de base de données.

Erreurs

- DBSubnetGroupAlreadyExistsFault
- DBSubnetGroupQuotaExceededFault
- DBSubnetQuotaExceededFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

Supprimer le DBSubnet groupe (action)

Le nom de la AWS CLI pour cette API est :`delete-db-subnet-group`.

Supprime un groupe de sous-réseaux de base de données.

> **Note**
>
> Le groupe de sous-réseaux de base de données spécifié ne doit pas être associé à des instances de base de données.

Demande

- DBSubnetGroupName (dans la CLI : `--db-subnet-group-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de sous-réseaux de base de données à supprimer.

  > **Note**
  >
  > Vous ne pouvez pas supprimer le groupe de sous-réseaux par défaut.

  Contraintes :

  Contraintes : Doit correspondre au nom d'un DBSubnet groupe existant. Impossible de conserver le nom par défaut.

  Exemple : `mySubnetgroup`

Réponse

- Paramètres d'absence de réponse.

Erreurs

- Non valide DBSubnet GroupStateFault
- Non valide DBSubnet StateFault

- DBSubnetGroupNotFoundFault

Modifier DBSubnet le groupe (action)

Le nom de la AWS CLI pour cette API est :`modify-db-subnet-group`.

Modifie un groupe de sous-réseaux de base de données existant. Les groupes de sous-réseaux de base de données doivent contenir au moins un sous-réseau sur au moins deux AZs dans la région Amazon.

Demande

- DBSubnetGroupDescription (dans la CLI : `--db-subnet-group-description`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Description du groupe de sous-réseaux de base de données.
- DBSubnetGroupName (dans la CLI : `--db-subnet-group-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de sous-réseaux de base de données. Cette valeur est stockée sous la forme d'une chaîne en minuscules. Vous ne pouvez pas modifier le groupe de sous-réseaux par défaut.

  Contraintes : Doit correspondre au nom d'un DBSubnet groupe existant. Impossible de conserver le nom par défaut.

  Exemple : `mySubnetgroup`
- SubnetIds (dans la CLI : `--subnet-ids`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le EC2 sous-réseau du groupe IDs de sous-réseaux de base de données.

Réponse

Contient les détails d'un groupe de sous-réseaux de base de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called “Décrire DBSubnet les groupes”.

- DBSubnetGroupArn : chaîne de type : `string` (chaîne encodée en UTF-8).

Amazon Resource Name (ARN) du groupe de sous-réseaux de base de données.

- DBSubnetGroupDescription : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la description du groupe de sous-réseaux de base de données.

- DBSubnetGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le nom du groupe de sous-réseaux DB.

- SubnetGroupStatus : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le statut du groupe de sous-réseaux de base de données.

- Subnets : tableau d'objets Sous-réseau.

  Contient une liste des éléments the section called "Sous-réseau".

- VpcId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom VpcId du groupe de sous-réseaux de base de données.

Erreurs

- DBSubnetGroupNotFoundFault
- DBSubnetQuotaExceededFault
- SubnetAlreadyInUse
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

Décrire DBSubnet les groupes (action)

Le nom de la AWS CLI pour cette API est :`describe-db-subnet-groups`.

Renvoie une liste de descriptions de DBSubnet groupes. Si a DBSubnet GroupName est spécifié, la liste contiendra uniquement les descriptions du DBSubnet groupe spécifié.

Pour obtenir une vue d'ensemble des plages d'adresses CIDR, consultez le didacticiel Wikipedia.

Demande

- DBSubnetGroupName (dans la CLI : `--db-subnet-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de sous-réseaux de base de données pour lequel renvoyer les détails.

- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Ce paramètre n'est actuellement pas pris en charge.

- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Un jeton de pagination facultatif fourni par une précédente demande Describe DBSubnet Groups. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.

Réponse

- DBSubnetGroups : tableau d'objets DBSubnetGroupe.

  Liste des instances the section called "DBSubnetGroupe".

- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

Erreurs

- DBSubnetGroupNotFoundFault

Structures :

Subnet (structure)

Spécifie un sous-réseau.

Ce type de données est utilisé comme élément de réponse dans l'action the section called "Décrire DBSubnet les groupes".

Champs

- SubnetAvailabilityZone : objet AvailabilityZone.

  Spécifie la zone de EC2 disponibilité dans laquelle se trouve le sous-réseau.
- SubnetIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'identifiant du sous-réseau.
- SubnetStatus : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le statut du sous-réseau.

DBSubnetGroupe (structure)

Contient les détails d'un groupe de sous-réseaux de base de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called "Décrire DBSubnet les groupes".

Champs

- DBSubnetGroupArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du groupe de sous-réseaux de base de données.
- DBSubnetGroupDescription : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la description du groupe de sous-réseaux de base de données.
- DBSubnetGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le nom du groupe de sous-réseaux DB.

- SubnetGroupStatus : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le statut du groupe de sous-réseaux de base de données.

- Subnets : tableau d'objets Sous-réseau.

  Contient une liste des éléments the section called "Sous-réseau".

- VpcId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom VpcId du groupe de sous-réseaux de base de données.

`DBSubnetGroup` est utilisé comme élément de réponse pour :

- Créer un DBSubnet groupe
- Modifier le DBSubnet groupe

API d'instantanés Neptune

Actions :

- Créer DBCluster un instantané (action)
- Supprimer DBCluster un instantané (action)
- Copier DBCluster un instantané (action)
- Modifier DBCluster SnapshotAttribute (action)
- Restaurer DBCluster FromSnapshot (action)
- Restaurer DBCluster ToPointInTime (action)
- Décrire les DBCluster instantanés (action)
- Décrire DBCluster SnapshotAttributes (action)

Structures :

- DBClusterInstantané (structure)
- DBClusterSnapshotAttribute (structure)
- DBClusterSnapshotAttributesResult (structure)

Créer DBCluster un instantané (action)

Le nom de la AWS CLI pour cette API est :`create-db-cluster-snapshot`.

Crée un instantané d'un cluster de bases de données.

Demande

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du cluster de bases de données pour lequel créer un instantané. Ce paramètre n'est pas sensible à la casse.

  Contraintes :

  - Doit correspondre à l'identifiant d'un existant DBCluster.

  Exemple : `my-cluster1`
- DBClusterSnapshotIdentifier (dans la CLI : `--db-cluster-snapshot-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de l'instantané de cluster de bases de données. Ce paramètre est stocké sous la forme d'une chaîne en lettres minuscules.

  Contraintes :

  - Doit contenir entre 1 et 63 lettres, chiffres ou traits d'union.
  - Le premier caractère doit être une lettre.
  - Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs.

  Exemple : `my-cluster1-snapshot1`
- Tags (dans la CLI : `--tags`) : tableau d'objets Tag.

  Balises à attribuer à l'instantané de cluster de bases de données.

Réponse

Contient les détails d'un instantané de cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called “Décrire les DBCluster instantanés”.

- AllocatedStorage : entier de type : `integer` (entier signé de 32 bits).

  Spécifie la taille de stockage allouée en gigaoctets (Gio).

- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du snapshot du cluster de base de données peuvent être restaurées.

- ClusterCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'identifiant du cluster de bases de données à partir duquel cet instantané de cluster de bases de données a été créé.

- DBClusterSnapshotArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'instantané de cluster de bases de données.

- DBClusterSnapshotIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'identifiant d'un instantané de cluster de bases de données. Doit correspondre à l'identifiant d'un instantané existant.

  Après avoir restauré un cluster de bases de données à l'aide d'un `DBClusterSnapshotIdentifier`, vous devez spécifier le même `DBClusterSnapshotIdentifier` pour toute mise à jour ultérieure de ce cluster. Lorsque vous spécifiez cette propriété pour une mise à jour, le cluster de bases de données n'est pas restauré de nouveau à partir de l'instantané, et les données de la base de données ne sont pas modifiées.

  Toutefois, si vous ne spécifiez pas le `DBClusterSnapshotIdentifier`, un cluster de bases de données vide est créé et le cluster de bases de données d'origine est supprimé. Si vous spécifiez une propriété différente de la propriété précédente de restauration d'instantané, le cluster de bases de données est restauré à partir de l'instantané spécifié par le `DBClusterSnapshotIdentifier`, et le cluster de bases de données d'origine est supprimé.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du moteur de base de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la version du moteur de base de données à utiliser pour cet instantané de cluster de bases de données.
- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).
- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), il s'agit de l'identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.
- LicenseModel : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit les informations sur le modèle de licence pour cet instantané de cluster de bases de données.
- PercentProgress : entier de type : `integer` (entier signé de 32 bits).

  Spécifie une estimation du pourcentage de données transférées.
- Port : entier de type : `integer` (entier signé de 32 bits).

  Spécifie le port écouté par le cluster de bases de données au moment de l'instantané.
- SnapshotCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle l'instantané a été pris, au format UTC (temps universel).
- SnapshotType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le type de l'instantané de cluster de bases de données.
- SourceDBClusterSnapshotArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si l'instantané de cluster de bases de données a été copié à partir d'un instantané de cluster de bases de données source, l'Amazon Resource Name (ARN) de l'instantané de cluster de bases de données source, une valeur null dans le cas contraire.
- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le statut de cet instantané de cluster de bases de données.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si l'instantané de cluster de bases de données est chiffré.
- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de stockage associé à l'instantané de cluster de bases de données.
- VpcId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit l'ID de VPC associé à l'instantané de cluster de bases de données.

Erreurs

- DBClusterSnapshotAlreadyExistsFault
- Non valide DBCluster StateFault
- DBClusterNotFoundFault
- SnapshotQuotaExceededFault
- Non valide DBCluster SnapshotStateFault

Supprimer DBCluster un instantané (action)

Le nom de la AWS CLI pour cette API est :`delete-db-cluster-snapshot`.

Supprime un instantané de cluster de bases de données. Si l'instantané est en cours de copie, l'opération est arrêtée.

> **Note**
>
> L'instantané de cluster de bases de données doit être dans l'état `available` pour être supprimé.

Demande

- DBClusterSnapshotIdentifier (dans la CLI : `--db-cluster-snapshot-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de l'instantané de cluster de bases de données à supprimer.

Contraintes : doit être le nom d'un instantané de cluster de bases de données existant dans l'état `available`.

Réponse

Contient les détails d'un instantané de cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called "Décrire les DBCluster instantanés".

- AllocatedStorage : entier de type : `integer` (entier signé de 32 bits).

  Spécifie la taille de stockage allouée en gigaoctets (Gio).

- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du snapshot du cluster de base de données peuvent être restaurées.

- ClusterCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'identifiant du cluster de bases de données à partir duquel cet instantané de cluster de bases de données a été créé.

- DBClusterSnapshotArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'instantané de cluster de bases de données.

- DBClusterSnapshotIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'identifiant d'un instantané de cluster de bases de données. Doit correspondre à l'identifiant d'un instantané existant.

  Après avoir restauré un cluster de bases de données à l'aide d'un `DBClusterSnapshotIdentifier`, vous devez spécifier le même `DBClusterSnapshotIdentifier` pour toute mise à jour ultérieure de ce cluster. Lorsque vous

spécifiez cette propriété pour une mise à jour, le cluster de bases de données n'est pas restauré de nouveau à partir de l'instantané, et les données de la base de données ne sont pas modifiées.

Toutefois, si vous ne spécifiez pas le `DBClusterSnapshotIdentifier`, un cluster de bases de données vide est créé et le cluster de bases de données d'origine est supprimé. Si vous spécifiez une propriété différente de la propriété précédente de restauration d'instantané, le cluster de bases de données est restauré à partir de l'instantané spécifié par le `DBClusterSnapshotIdentifier`, et le cluster de bases de données d'origine est supprimé.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du moteur de base de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la version du moteur de base de données à utiliser pour cet instantané de cluster de bases de données.

- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), il s'agit de l'identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.

- LicenseModel : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit les informations sur le modèle de licence pour cet instantané de cluster de bases de données.

- PercentProgress : entier de type : `integer` (entier signé de 32 bits).

  Spécifie une estimation du pourcentage de données transférées.

- Port : entier de type : `integer` (entier signé de 32 bits).

  Spécifie le port écouté par le cluster de bases de données au moment de l'instantané.

- SnapshotCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle l'instantané a été pris, au format UTC (temps universel).

- SnapshotType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le type de l'instantané de cluster de bases de données.

- SourceDBClusterSnapshotArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si l'instantané de cluster de bases de données a été copié à partir d'un instantané de cluster de bases de données source, l'Amazon Resource Name (ARN) de l'instantané de cluster de bases de données source, une valeur null dans le cas contraire.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le statut de cet instantané de cluster de bases de données.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si l'instantané de cluster de bases de données est chiffré.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de stockage associé à l'instantané de cluster de bases de données.

- VpcId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit l'ID de VPC associé à l'instantané de cluster de bases de données.

Erreurs

- Non valide DBCluster SnapshotStateFault
- DBClusterSnapshotNotFoundFault

Copier DBCluster un instantané (action)

Le nom de la AWS CLI pour cette API est :`copy-db-cluster-snapshot`.

Copie un instantané d'un cluster de bases de données.

Pour copier un instantané de cluster de bases de données à partir d'un instantané de cluster de bases de données manuel partagé, `SourceDBClusterSnapshotIdentifier` doit être l'ARN (Amazon Resource Name) de l'instantané de cluster de bases de données partagé.

Demande

- CopyTags(dans la CLI :`--copy-tags`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  La valeur est true pour copier toutes les balises de l'instantané de cluster de bases de données source vers l'instantané de cluster de bases de données cible, et false dans le cas contraire. La valeur par défaut est false.

- KmsKeyId (dans la CLI : `--kms-key-id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de clé Amazon KMS pour un instantané de cluster de bases de données chiffré. L'ID de clé KMS est l'Amazon Resource Name (ARN), l'identifiant de clé KMS ou l'alias de clé KMS pour la clé de chiffrement KMS.

  Si vous copiez un instantané de cluster de bases de données chiffré à partir de votre compte Amazon, vous pouvez spécifier une valeur pour `KmsKeyId` afin de chiffrer la copie avec une nouvelle clé de chiffrement KMS. Si vous ne spécifiez pas de valeur pour `KmsKeyId`, la copie de l'instantané de cluster de bases de données est chiffrée avec la même clé KMS que l'instantané de cluster de bases de données source.

  Si vous copiez un instantané de cluster de bases de données chiffré partagé à partir d'un autre compte Amazon, vous devez spécifier une valeur pour `KmsKeyId`.

  Les clés de chiffrement KMS sont spécifiques à la région Amazon dans laquelle elles sont créées, et vous ne pouvez pas utiliser de clés de chiffrement issues d'une région Amazon dans une autre région Amazon.

  Vous ne pouvez pas chiffrer un instantané de cluster de bases de données non chiffré lorsque vous le copiez. Si vous essayez de copier un instantané de cluster de base de données non chiffré et que vous spécifiez une valeur pour le KmsKeyId paramètre, une erreur est renvoyée.

- PreSignedUrl (dans la CLI : `--pre-signed-url`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Cette option n'est pas prise en charge actuellement.

- SourceDBClusterSnapshotIdentifier (dans la CLI : `--source-db-cluster-snapshot-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de l'instantané de cluster de bases de données à copier. Ce paramètre n'est pas sensible à la casse.

  Contraintes :

- Vous devez spécifier un instantané système valide dans l'état « available ».
- Spécifiez un identifiant de l'instantané de base de données valide.

  Exemple : `my-cluster-snapshot1`

- Tags (dans la CLI : `--tags`) : tableau d'objets Tag.

  Balises à attribuer à la nouvelle copie de l'instantané de cluster de bases de données.

- TargetDBClusterSnapshotIdentifier (dans la CLI : `--target-db-cluster-snapshot-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du nouvel instantané de cluster de bases de données à créer à partir de l'instantané de cluster de bases de données source. Ce paramètre n'est pas sensible à la casse.

  Contraintes :

  - Doit contenir entre 1 et 63 lettres, chiffres ou traits d'union.
  - Le premier caractère doit être une lettre.
  - Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs.

  Exemple : `my-cluster-snapshot2`

Réponse

Contient les détails d'un instantané de cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called “Décrire les DBCluster instantanés”.

- AllocatedStorage : entier de type : `integer` (entier signé de 32 bits).

  Spécifie la taille de stockage allouée en gigaoctets (Gio).

- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du snapshot du cluster de base de données peuvent être restaurées.

- ClusterCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'identifiant du cluster de bases de données à partir duquel cet instantané de cluster de bases de données a été créé.

- DBClusterSnapshotArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'instantané de cluster de bases de données.

- DBClusterSnapshotIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'identifiant d'un instantané de cluster de bases de données. Doit correspondre à l'identifiant d'un instantané existant.

  Après avoir restauré un cluster de bases de données à l'aide d'un `DBClusterSnapshotIdentifier`, vous devez spécifier le même `DBClusterSnapshotIdentifier` pour toute mise à jour ultérieure de ce cluster. Lorsque vous spécifiez cette propriété pour une mise à jour, le cluster de bases de données n'est pas restauré de nouveau à partir de l'instantané, et les données de la base de données ne sont pas modifiées.

  Toutefois, si vous ne spécifiez pas le `DBClusterSnapshotIdentifier`, un cluster de bases de données vide est créé et le cluster de bases de données d'origine est supprimé. Si vous spécifiez une propriété différente de la propriété précédente de restauration d'instantané, le cluster de bases de données est restauré à partir de l'instantané spécifié par le `DBClusterSnapshotIdentifier`, et le cluster de bases de données d'origine est supprimé.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du moteur de base de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la version du moteur de base de données à utiliser pour cet instantané de cluster de bases de données.

- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), il s'agit de l'identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.
- LicenseModel : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit les informations sur le modèle de licence pour cet instantané de cluster de bases de données.
- PercentProgress : entier de type : `integer` (entier signé de 32 bits).

  Spécifie une estimation du pourcentage de données transférées.
- Port : entier de type : `integer` (entier signé de 32 bits).

  Spécifie le port écouté par le cluster de bases de données au moment de l'instantané.
- SnapshotCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle l'instantané a été pris, au format UTC (temps universel).
- SnapshotType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le type de l'instantané de cluster de bases de données.
- SourceDBClusterSnapshotArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si l'instantané de cluster de bases de données a été copié à partir d'un instantané de cluster de bases de données source, l'Amazon Resource Name (ARN) de l'instantané de cluster de bases de données source, une valeur null dans le cas contraire.
- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le statut de cet instantané de cluster de bases de données.
- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si l'instantané de cluster de bases de données est chiffré.
- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de stockage associé à l'instantané de cluster de bases de données.
- VpcId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit l'ID de VPC associé à l'instantané de cluster de bases de données.

Erreurs

- DBClusterSnapshotAlreadyExistsFault
- DBClusterSnapshotNotFoundFault
- Non valide DBCluster StateFault
- Non valide DBCluster SnapshotStateFault
- SnapshotQuotaExceededFault
- KMSKeyNotAccessibleFault

Modifier DBCluster SnapshotAttribute (action)

Le nom de la AWS CLI pour cette API est :`modify-db-cluster-snapshot-attribute`.

Ajoute un attribut et des valeurs à un instantané de cluster de bases de données manuel, ou en supprime.

Pour partager un instantané de cluster de base de données manuel avec d'autres comptes Amazon, spécifiez `restore` comme `ValuesToAdd` paramètre `AttributeName` et utilisez le paramètre pour ajouter une liste IDs des comptes Amazon autorisés à restaurer l'instantané de cluster de base de données manuel. Utilisez la valeur `all` pour rendre l'instantané de cluster de bases de données manuel public, ce qui signifie qu'il pourra être copié ou restauré par tous les comptes Amazon. N'ajoutez pas la valeur `all` pour tous les instantanés de cluster de bases de données manuels contenant des informations privées que vous ne souhaitez pas mettre à la disposition de tous les comptes Amazon. Si un instantané de cluster de base de données manuel est chiffré, il peut être partagé, mais uniquement en spécifiant une liste de comptes Amazon autorisés IDs pour le `ValuesToAdd` paramètre. Dans ce cas, vous ne pouvez pas utiliser `all` comme une valeur pour ce paramètre.

Pour afficher les comptes Amazon autorisés à copier ou restaurer un instantané de cluster de bases de données manuel, ou pour voir si un instantané de cluster de bases de données manuel est public ou privé, utilisez l'action d'API the section called “Décrivez DBCluster SnapshotAttributes”.

Demande

- AttributeName (dans la CLI : `--attribute-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de l'attribut d'instantané de cluster de bases de données à modifier.

Pour gérer l'autorisation des autres comptes Amazon à copier ou restaurer un instantané de cluster de bases de données manuel, définissez cette valeur sur `restore`.

- DBClusterSnapshotIdentifier (dans la CLI : `--db-cluster-snapshot-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de l'instantané de cluster de bases de données pour lequel modifier les attributs.

- ValuesToAdd (dans la CLI : `--values-to-add`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des attributs d'instantané de cluster de bases de données à ajouter à l'attribut spécifié par `AttributeName`.

  Pour autoriser d'autres comptes Amazon à copier ou à restaurer un instantané de cluster de base de données manuel, configurez cette liste pour inclure un ou plusieurs comptes Amazon IDs, ou `all` pour que l'instantané de cluster de base de données manuel puisse être restauré par n'importe quel compte Amazon. N'ajoutez pas la valeur `all` pour tous les instantanés de cluster de bases de données manuels contenant des informations privées que vous ne souhaitez pas mettre à la disposition de tous les comptes Amazon.

- ValuesToRemove (dans la CLI : `--values-to-remove`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des attributs d'instantané de cluster de bases de données à supprimer de l'attribut spécifié par `AttributeName`.

  Pour supprimer l'autorisation d'autres comptes Amazon à copier ou restaurer un instantané de cluster de bases de données manuel, définissez cette liste afin d'inclure un ou plusieurs identifiants de compte Amazon, ou définissez `all` pour interdire à tous les comptes Amazon de copier ou restaurer l'instantané de cluster de bases de données. Si vous spécifiez `all`, un compte Amazon dont l'ID de compte est explicitement ajouté à l'attribut `restore` peut toujours copier ou restaurer un instantané de cluster de bases de données manuel.

Réponse

Contient les résultats d'appel réussi à l'action d'API the section called “Décrivez DBCluster SnapshotAttributes”.

Les attributs d'instantané de cluster de bases de données manuel sont utilisés pour autoriser d'autres comptes Amazon à copier ou restaurer un instantané de cluster de bases de données manuel. Pour plus d'informations, consultez l'action d'API the section called "Modifier DBCluster SnapshotAttribute".

- DBClusterSnapshotAttributes : tableau d'objets DBClusterSnapshotAttribute.

  Liste des attributs et des valeurs de l'instantané de cluster de bases de données manuel.
- DBClusterSnapshotIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de l'instantané de cluster de bases de données manuel sur lequel s'appliquent les attributs.

Erreurs

- DBClusterSnapshotNotFoundFault
- Non valide DBCluster SnapshotStateFault
- SharedSnapshotQuotaExceededFault

Restaurer DBCluster FromSnapshot (action)

Le nom de la AWS CLI pour cette API est :`restore-db-cluster-from-snapshot`.

Crée un nouveau cluster de bases de données à partir d'un instantané de base de données ou d'un instantané de cluster de bases de données.

Si un instantané de base de données est spécifié, le cluster de bases de données cible est créé à partir de l'instantané de base de données source avec une configuration par défaut et un groupe de sécurité par défaut.

Si un instantané de cluster de bases de données est spécifié, le cluster de bases de données cible est créé à partir du point de restauration du cluster de bases de données source avec la même configuration que le cluster de bases de données source original, sauf que le nouveau cluster de bases de données est créé avec le groupe de sécurité par défaut.

Demande

- AvailabilityZones (dans la CLI : `--availability-zones`) : chaîne de type : `string` (chaîne encodée en UTF-8).

Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du cluster de base de données restauré peuvent être créées.

- CopyTagsToSnapshot(dans la CLI :`--copy-tags-to-snapshot`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est *`true`*, les balises sont copiées dans n'importe quel instantané du cluster de bases de données restauré qui est créé.

- DatabaseName (dans la CLI : `--database-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge.

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du cluster de bases de données à créer à partir de l'instantané de base de données ou de l'instantané de cluster de bases de données. Ce paramètre n'est pas sensible à la casse.

  Contraintes :

  - Doit contenir entre 1 et 63 lettres, chiffres ou traits d'union
  - Le premier caractère doit être une lettre
  - Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs

  Exemple : `my-snapshot-id`

- DBClusterParameterGroupName (dans la CLI : `--db-cluster-parameter-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de paramètres de cluster de base de données à associer au nouveau cluster de base de données.

  Contraintes :

  - S'il est fourni, il doit correspondre au nom d'un existant DBClusterParameterGroup.

- DBSubnetGroupName (dans la CLI : `--db-subnet-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de sous-réseaux de base de données à utiliser pour le nouveau cluster de bases de données.

Contraintes : s'il est indiqué, il doit correspondre au nom d'un DBSubnet groupe existant.

Exemple : `mySubnetgroup`

- DeletionProtection(dans la CLI :`--deletion-protection`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Valeur qui indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée. Par défaut, la protection contre la suppression est désactivée.
- EnableCloudwatchLogsExports (dans la CLI : `--enable-cloudwatch-logs-exports`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des journaux que le cluster de base de données restauré doit exporter vers Amazon CloudWatch Logs.
- EnableIAMDatabaseAuthentication(dans la CLI :`--enable-iam-database-authentication`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Valeur true pour activer le mappage des comptes Amazon Identity and Access Management (IAM) avec les comptes de base de données ; sinon, valeur false.

  Par défaut : `false`
- Engine (dans la CLI : `--engine`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Moteur de base de données à utiliser pour le nouveau cluster de base de données.

  Par défaut : identique à la source

  Contrainte : doit être compatible avec le moteur de la source
- EngineVersion (dans la CLI : `--engine-version`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Version du moteur de base de données à utiliser pour le nouveau cluster de bases de données.
- KmsKeyId (dans la CLI : `--kms-key-id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de clé Amazon KMS à utiliser lors de la restauration d'un cluster de bases de données chiffré à partir d'un instantané de base de données ou d'un instantané de cluster de bases de données.

L'identifiant de clé KMS est l'Amazon Resource Name (ARN) de la clé de chiffrement KMS. Si vous restaurez un cluster de bases de données avec le compte Amazon qui possède la clé de chiffrement KMS utilisée pour chiffrer le nouveau cluster de bases de données, vous pouvez utiliser l'alias de clé KMS au lieu de l'ARN de la clé de chiffrement KMS.

Si vous ne spécifiez pas de valeur pour le paramètre `KmsKeyId` :

- Si l'instantané de base de données ou l'instantané de cluster de bases de données dans `SnapshotIdentifier` est chiffré, le cluster de bases de données restauré est chiffré à l'aide de la clé KMS utilisée pour chiffrer l'instantané de base de données ou l'instantané de cluster de bases de données.
- Si l'instantané de base de données ou l'instantané de cluster de bases de données dans `SnapshotIdentifier` n'est pas chiffré, le cluster de base de données restauré n'est pas chiffré.

- Port(dans la CLI :`--port`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Le numéro de port sur lequel le nouveau cluster de bases de données accepte des connexions.

  Contraintes : La valeur doit être comprise dans la plage `1150-65535`

  Valeur par défaut : port du cluster de bases de données d'origine.

- ServerlessV2ScalingConfiguration (dans la CLI : `--serverless-v2-scaling-configuration`) : objet Serverless V2 ScalingConfiguration.

  Contient la configuration de mise à l'échelle d'un cluster de bases de données Neptune sans serveur.

  Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

- SnapshotIdentifier (dans la CLI : `--snapshot-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de l'instantané de base de données ou de l'instantané de cluster de bases de données à partir duquel restaurer.

  Vous pouvez utiliser le nom ou l'Amazon Resource Name (ARN) pour spécifier un instantané de cluster de bases de données. Cependant, vous pouvez utiliser uniquement l'ARN pour spécifier un instantané de base de données.

Contraintes :

- Doit correspondre à l'identifiant d'un instantané existant.

- StorageType (dans la CLI : `--storage-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le type de stockage à associer au cluster de base de données.

  Valeurs valides : `standard`, `iopt1`

  Par défaut : `standard`

- Tags (dans la CLI : `--tags`) : tableau d'objets Tag.

  Balises à attribuer au cluster de bases de données restauré.

- VpcSecurityGroupIds (dans la CLI : `--vpc-security-group-ids`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des groupes de sécurité VPC auxquels appartiendra le nouveau cluster de bases de données.

Réponse

Contient les détails d'un cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans the section called "Décrivez DBClusters".

- AllocatedStorage— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  `AllocatedStorage` renvoie toujours la valeur 1, car la taille de stockage d'un cluster de bases de données Neptune n'est pas fixe ; elle s'ajuste automatiquement en fonction des besoins.

- AssociatedRoles : tableau d'objets DBClusterRôle.

  Fournit la liste des rôles Amazon Identity and Access Management (IAM) associés au cluster de bases de données. Les rôles IAM associés à un cluster de bases de données autorisent celui-ci à accéder aux autres services Amazon en votre nom.

- AutomaticRestartTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

Heure à laquelle le cluster de bases de données sera automatiquement redémarré.

- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du cluster de base de données peuvent être créées.

- BacktrackConsumedChangeRecords— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.

- BacktrackWindow— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.

- BackupRetentionPeriod— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.

- Capacity— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Non pris en charge par Neptune.

- CloneGroupId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifie le groupe de clones auquel est associé le cluster de bases de données.

- ClusterCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).

- CopyTagsToSnapshot— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur *`true`*, les balises sont copiées dans n'importe quel instantané du cluster de bases de données créé.

- CrossAccountClone— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur `true`, le cluster de bases de données peut être cloné sur plusieurs comptes.

- DatabaseName : chaîne de type : `string` (chaîne encodée en UTF-8).

Contient le nom de la base de données initiale de ce cluster de bases de données qui a été fourni au moment de la création, dans la mesure où un nom a été spécifié au moment de créer le cluster de bases de données. Ce même nom est renvoyé pendant la durée de vie du cluster de bases de données.

- DBClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du cluster de bases de données.

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de cluster de bases de données fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie un cluster de bases de données.

- DBClusterMembers : tableau d'objets DBClusterMembre.

  Fournit la liste des instances qui composent le cluster de bases de données.

- DBClusterParameterGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du groupe de paramètres de cluster de base de données pour le cluster de base de données.

- DbClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à la région Amazon du cluster de bases de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS du cluster de base de données est consultée.

- DBSubnetGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie les informations sur le groupe de sous-réseaux associé au cluster de bases de données, notamment le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée.

- EarliestBacktrackTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Non pris en charge par Neptune.

- EarliestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie la première heure à laquelle une base de données peut être restaurée par point-in-time restauration.

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux que ce cluster de base de données est configuré pour exporter vers CloudWatch Logs. Les types de journaux valides sont les suivants : `audit` (pour publier les journaux d'audit CloudWatch) et slowquery (pour publier les journaux de requêtes lentes). CloudWatch Consultez la section Publication des journaux Neptune sur Amazon CloudWatch Logs.

- Endpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le point de terminaison de connexion pour l'instance principale du cluster de bases de données.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour ce cluster de bases de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.

- HostedZoneId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'ID attribué par Amazon Route 53 lorsque vous créez une zone hébergée.

- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).

- IOOptimizedNextAllowedModificationTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

La prochaine fois, vous pourrez modifier le cluster de bases de données de façon à utiliser le type de stockage `iopt1`.

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.

- LatestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée par point-in-time restauration.

- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de bases de données possède des instances dans plusieurs zones de disponibilité.

- PendingModifiedValues : objet ClusterPendingModifiedValues.

  Ce type de données est utilisé comme élément de réponse dans l'opération `ModifyDBCluster` et contient les modifications qui seront appliquées lors de la fenêtre de maintenance suivante.

- PercentProgress : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la progression de l'opération sous forme de pourcentage.

- Port— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le port sur lequel le moteur de base de données est à l'écoute.

- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.

- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.

- ReaderEndpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

Point de terminaison du lecteur pour le cluster de bases de données. Le point de terminaison du lecteur d'un cluster de bases de données équilibre la charge des connexions entre les réplicas en lecture qui sont disponibles dans un cluster de bases de données. À mesure que les clients demandent de nouvelles connexions au point de terminaison du lecteur, Neptune répartit les demandes de connexion entre les réplicas en lecture du cluster de bases de données. Cette fonctionnalité peut contribuer à équilibrer votre charge de travail entre les différents réplicas en lecture de votre cluster de bases de données.

Si un basculement se produit et que le réplica en lecture auquel vous êtes connecté est promu en instance principale, votre connexion est supprimée. Pour continuer à envoyer votre charge de travail de lecture à d'autres réplicas en lecture du cluster, vous pouvez alors vous reconnecter au point de terminaison du lecteur.

- ReadReplicaIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à ce cluster de bases de données.

- ReplicationSourceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ReplicationType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ServerlessV2ScalingConfiguration : objet Serverless V2 ScalingConfigurationInfo.

  Affiche la configuration de mise à l'échelle pour un cluster de bases de données Neptune sans serveur.

  Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'état actuel de ce cluster de base de données.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de base de données est chiffré.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

Type de stockage utilisé par le cluster de bases de données.

Valeurs valides :

- **`standard`**— (valeur par défaut) Fournit un stockage de base de données économique pour les applications dont l' I/O utilisation est modérée à faible.
- **`iopt1`**— Permet un stockage optimisé pour les E/S, conçu pour répondre aux besoins de I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O débit.

  Le stockage optimisé pour les E/S Neptune n'est disponible qu'à partir de la version 1.3.0.0 du moteur.

- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit la liste des groupes de sécurité VPC auxquels appartient le cluster de base de données.

Erreurs

- DBClusterAlreadyExistsFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterSnapshotNotFoundFault
- Insuffisant DBCluster CapacityFault
- InsufficientStorageClusterCapacityFault
- Non valide DBSnapshot StateFault
- Non valide DBCluster SnapshotStateFault
- StorageQuotaExceededFault
- Non valide VPCNetwork StateFault
- InvalidRestoreFault
- DBSubnetGroupNotFoundFault
- InvalidSubnet
- OptionGroupNotFoundFault

- KMSKeyNotAccessibleFault
- DBClusterParameterGroupNotFoundFault

Restaurer DBCluster ToPointInTime (action)

Le nom de la AWS CLI pour cette API est :`restore-db-cluster-to-point-in-time`.

Restaure un cluster de bases de données à un moment donné arbitraire. Les utilisateurs peuvent restaurer à tout moment avant `LatestRestorableTime` pendant `BackupRetentionPeriod` jours. Le cluster de bases de données cible est créé à partir du cluster de bases de données source avec la même configuration que le cluster de bases de données original, sauf que le nouveau cluster de bases de données est créé avec le groupe de sécurité de base de données par défaut.

> **Note**
>
> Cette action restaure uniquement le cluster de bases de données, mais pas les instances de base de données de ce cluster de bases de données. Vous devez invoquer l'action the section called "Créez DBInstance" pour créer des instances de bases de données pour le cluster de bases de données restauré, en spécifiant l'identifiant du cluster de bases de données restauré dans `DBClusterIdentifier`. Vous pouvez créer des instances de base de données uniquement après la fin de l'action `RestoreDBClusterToPointInTime` et lorsque le cluster de bases de données est disponible.

Demande

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du nouveau cluster de bases de données à créer.

  Contraintes :

  - Doit contenir entre 1 et 63 lettres, chiffres ou traits d'union
  - Le premier caractère doit être une lettre
  - Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs
- DBClusterParameterGroupName (dans la CLI : `--db-cluster-parameter-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

Nom du groupe de paramètres de cluster de base de données à associer au nouveau cluster de base de données.

Contraintes :

- S'il est fourni, il doit correspondre au nom d'un existant DBClusterParameterGroup.

- DBSubnetGroupName (dans la CLI : `--db-subnet-group-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de sous-réseaux de base de données à utiliser pour le nouveau cluster de bases de données.

  Contraintes : s'il est indiqué, il doit correspondre au nom d'un DBSubnet groupe existant.

  Exemple : `mySubnetgroup`

- DeletionProtection(dans la CLI :`--deletion-protection`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Valeur qui indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée. Par défaut, la protection contre la suppression est désactivée.

- EnableCloudwatchLogsExports (dans la CLI : `--enable-cloudwatch-logs-exports`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  La liste des journaux que le cluster de base de données restauré doit exporter vers CloudWatch Logs.

- EnableIAMDatabaseAuthentication(dans la CLI :`--enable-iam-database-authentication`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Valeur true pour activer le mappage des comptes Amazon Identity and Access Management (IAM) avec les comptes de base de données ; sinon, valeur false.

  Par défaut : `false`

- KmsKeyId (dans la CLI : `--kms-key-id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de clé Amazon KMS à utiliser lors de la restauration d'un cluster de bases de données chiffré à partir d'un cluster de bases de données chiffré.

L'identifiant de clé KMS est l'Amazon Resource Name (ARN) de la clé de chiffrement KMS. Si vous restaurez un cluster de bases de données avec le compte Amazon qui possède la clé de chiffrement KMS utilisée pour chiffrer le nouveau cluster de bases de données, vous pouvez utiliser l'alias de clé KMS au lieu de l'ARN de la clé de chiffrement KMS.

Vous pouvez restaurer vers un nouveau cluster de bases de données et chiffrer le nouveau cluster de bases de données avec une clé KMS différente de la clé KMS utilisée pour chiffrer le cluster de bases de données source. Le nouveau cluster de bases de données est chiffré avec la clé KMS identifiée par le paramètre `KmsKeyId`.

Si vous ne spécifiez pas de valeur pour le paramètre `KmsKeyId` :

- Si le cluster de bases de données est chiffré, le cluster de bases de données restauré est chiffré à l'aide de la clé KMS utilisée pour chiffrer le cluster de bases de données source.
- Si le cluster de bases de données n'est pas chiffré, le cluster de bases de données restauré n'est pas chiffré.

Si `DBClusterIdentifier` fait référence à un cluster de base de données non chiffré, la demande de restauration est rejetée.

- Port(dans la CLI :`--port`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Le numéro de port sur lequel le nouveau cluster de bases de données accepte des connexions.

  Contraintes : La valeur doit être comprise dans la plage `1150-65535`

  Valeur par défaut : port du cluster de bases de données d'origine.

- RestoreToTime(dans la CLI :`--restore-to-time`) — a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Date et heure auxquelles restaurer le cluster de bases de données.

  Valeurs Valides : doit être une heure au format UTC (temps universel)

  Contraintes :

  - Doit se situer avant l'heure de restauration la plus récente pour l'instance de base de données.
  - Doit être indiqué si le paramètre `UseLatestRestorableTime` n'est pas fourni.
  - Ne peut pas être spécifié si le paramètre `UseLatestRestorableTime` est true.
  - Ne peut pas être spécifié si le paramètre `RestoreType` est `copy-on-write`.

Exemple : `2015-03-07T23:45:00Z`

- RestoreType (dans la CLI : `--restore-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de restauration à exécuter. Vous pouvez spécifier l'une des valeurs suivantes :

  - `full-copy` - Le nouveau cluster de base de données est restauré sous la forme d'une copie intégrale du cluster de base de données source.
  - `copy-on-write` - Le nouveau cluster de base de données est restauré sous la forme d'un clone du cluster de base de données source.

  Si vous ne spécifiez pas de valeur pour `RestoreType`, le nouveau cluster de base de données est restauré sous la forme d'une copie intégrale du cluster de base de données source.

- ServerlessV2ScalingConfiguration (dans la CLI : `--serverless-v2-scaling-configuration`) : objet Serverless V2 ScalingConfiguration.

  Contient la configuration de mise à l'échelle d'un cluster de bases de données Neptune sans serveur.

  Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

- SourceDBClusterIdentifier (dans la CLI : `--source-db-cluster-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du cluster de base de données source à partir duquel effectuer la restauration.

  Contraintes :

  - Doit correspondre à l'identifiant d'un existant DBCluster.

- StorageType (dans la CLI : `--storage-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le type de stockage à associer au cluster de base de données.

  Valeurs valides : `standard`, `iopt1`

  Par défaut : `standard`

- Tags (dans la CLI : `--tags`) : tableau d'objets Tag.

Balises à appliquer au cluster de bases de données restauré.

- UseLatestRestorableTime (dans la CLI : `--use-latest-restorable-time`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur défini sur `true` pour restaurer le cluster de bases de données à l'heure de sauvegarde de restauration la plus récente, et `false` dans le cas contraire.

  Par défaut : `false`

  Contraintes : ne doit pas être spécifié si le paramètre `RestoreToTime` est fourni.
- VpcSecurityGroupIds (dans la CLI : `--vpc-security-group-ids`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des groupes de sécurité VPC auxquels appartient le nouveau cluster de bases de données.

Réponse

Contient les détails d'un cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans the section called "Décrivez DBClusters".

- AllocatedStorage— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  `AllocatedStorage` renvoie toujours la valeur 1, car la taille de stockage d'un cluster de bases de données Neptune n'est pas fixe ; elle s'ajuste automatiquement en fonction des besoins.
- AssociatedRoles : tableau d'objets DBClusterRôle.

  Fournit la liste des rôles Amazon Identity and Access Management (IAM) associés au cluster de bases de données. Les rôles IAM associés à un cluster de bases de données autorisent celui-ci à accéder aux autres services Amazon en votre nom.
- AutomaticRestartTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Heure à laquelle le cluster de bases de données sera automatiquement redémarré.
- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du cluster de base de données peuvent être créées.

- BacktrackConsumedChangeRecords— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.

- BacktrackWindow— a LongOptional, de type : `long` (un entier signé de 64 bits).

  Non pris en charge par Neptune.

- BackupRetentionPeriod— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le nombre de jours de conservation des instantanés de base de données automatiques.

- Capacity— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Non pris en charge par Neptune.

- CloneGroupId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifie le groupe de clones auquel est associé le cluster de bases de données.

- ClusterCreateTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).

- CopyTagsToSnapshot— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur *`true`*, les balises sont copiées dans n'importe quel instantané du cluster de bases de données créé.

- CrossAccountClone— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si cette valeur est définie sur `true`, le cluster de bases de données peut être cloné sur plusieurs comptes.

- DatabaseName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient le nom de la base de données initiale de ce cluster de bases de données qui a été fourni au moment de la création, dans la mesure où un nom a été spécifié au moment de créer le cluster

de bases de données. Ce même nom est renvoyé pendant la durée de vie du cluster de bases de données.

- DBClusterArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) du cluster de bases de données.

- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un identifiant de cluster de bases de données fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie un cluster de bases de données.

- DBClusterMembers : tableau d'objets DBClusterMembre.

  Fournit la liste des instances qui composent le cluster de bases de données.

- DBClusterParameterGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du groupe de paramètres de cluster de base de données pour le cluster de base de données.

- DbClusterResourceId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant immuable et propre à la région Amazon du cluster de bases de données. Cet identifiant se trouve dans les entrées du CloudTrail journal Amazon chaque fois que la clé Amazon KMS du cluster de base de données est consultée.

- DBSubnetGroup : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie les informations sur le groupe de sous-réseaux associé au cluster de bases de données, notamment le nom, la description et les sous-réseaux du groupe de sous-réseaux.

- DeletionProtection— a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Indique si la protection contre la suppression est activée pour le cluster de bases de données. La base de données ne peut pas être supprimée lorsque la protection contre la suppression est activée.

- EarliestBacktrackTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Non pris en charge par Neptune.

- EarliestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

Spécifie la première heure à laquelle une base de données peut être restaurée par point-in-time restauration.

- EnabledCloudwatchLogsExports : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des types de journaux que ce cluster de base de données est configuré pour exporter vers CloudWatch Logs. Les types de journaux valides sont les suivants : `audit` (pour publier les journaux d'audit CloudWatch) et slowquery (pour publier les journaux de requêtes lentes). CloudWatch Consultez la section Publication des journaux Neptune sur Amazon CloudWatch Logs.
- Endpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le point de terminaison de connexion pour l'instance principale du cluster de bases de données.
- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le nom du moteur de base de données à utiliser pour ce cluster de bases de données.
- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique la version du moteur de base de données.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, de type : `string` (une chaîne codée en UTF-8), pas moins de 1 ou plus de 255 ? 1er ? s, correspondant à cette expression régulière :`[A-Za-z][0-9A-Za-z-:._]*`.

  Contient un identifiant de cluster de bases de données globale fourni par l'utilisateur. Cet identifiant est la clé unique qui identifie une base de données globale.
- HostedZoneId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'ID attribué par Amazon Route 53 lorsque vous créez une zone hébergée.
- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).
- IOOptimizedNextAllowedModificationTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  La prochaine fois, vous pourrez modifier le cluster de bases de données de façon à utiliser le type de stockage `iopt1`.

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.
- LatestRestorableTime— a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure limite à laquelle une base de données peut être restaurée par point-in-time restauration.
- MultiAZ : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de bases de données possède des instances dans plusieurs zones de disponibilité.
- PendingModifiedValues : objet ClusterPendingModifiedValues.

  Ce type de données est utilisé comme élément de réponse dans l'opération `ModifyDBCluster` et contient les modifications qui seront appliquées lors de la fenêtre de maintenance suivante.
- PercentProgress : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la progression de l'opération sous forme de pourcentage.
- Port— an IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Spécifie le port sur lequel le moteur de base de données est à l'écoute.
- PreferredBackupWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la plage de temps quotidienne au cours de laquelle des sauvegardes automatiques sont créées si cette fonctionnalité est activée, comme déterminé par la propriété `BackupRetentionPeriod`.
- PreferredMaintenanceWindow : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'intervalle de temps hebdomadaire, au format Universal Coordinated Time (UTC), pendant lequel a lieu la maintenance du système.
- ReaderEndpoint : chaîne de type : `string` (chaîne encodée en UTF-8).

  Point de terminaison du lecteur pour le cluster de bases de données. Le point de terminaison du lecteur d'un cluster de bases de données équilibre la charge des connexions entre les réplicas en lecture qui sont disponibles dans un cluster de bases de données. À mesure que les clients demandent de nouvelles connexions au point de terminaison du lecteur, Neptune répartit les

demandes de connexion entre les réplicas en lecture du cluster de bases de données. Cette fonctionnalité peut contribuer à équilibrer votre charge de travail entre les différents réplicas en lecture de votre cluster de bases de données.

Si un basculement se produit et que le réplica en lecture auquel vous êtes connecté est promu en instance principale, votre connexion est supprimée. Pour continuer à envoyer votre charge de travail de lecture à d'autres réplicas en lecture du cluster, vous pouvez alors vous reconnecter au point de terminaison du lecteur.

- ReadReplicaIdentifiers : chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient un ou plusieurs identifiants des réplicas en lecture associés à ce cluster de bases de données.

- ReplicationSourceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ReplicationType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Non pris en charge par Neptune.

- ServerlessV2ScalingConfiguration : objet Serverless V2 ScalingConfigurationInfo.

  Affiche la configuration de mise à l'échelle pour un cluster de bases de données Neptune sans serveur.

  Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'état actuel de ce cluster de base de données.

- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si le cluster de base de données est chiffré.

- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de stockage utilisé par le cluster de bases de données.

  Valeurs valides :

  - **`standard`**— (valeur par défaut) Fournit un stockage de base de données économique pour les applications dont l' I/O utilisation est modérée à faible.

- **`iopt1`**— Permet un stockage optimisé pour les E/S, conçu pour répondre aux besoins de I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O débit.

  Le stockage optimisé pour les E/S Neptune n'est disponible qu'à partir de la version 1.3.0.0 du moteur.
- VpcSecurityGroups : tableau d'objets VpcSecurityGroupMembership.

  Fournit la liste des groupes de sécurité VPC auxquels appartient le cluster de base de données.

Erreurs

- DBClusterAlreadyExistsFault
- DBClusterNotFoundFault
- DBClusterQuotaExceededFault
- DBClusterSnapshotNotFoundFault
- DBSubnetGroupNotFoundFault
- Insuffisant DBCluster CapacityFault
- InsufficientStorageClusterCapacityFault
- Non valide DBCluster SnapshotStateFault
- Non valide DBCluster StateFault
- Non valide DBSnapshot StateFault
- InvalidRestoreFault
- InvalidSubnet
- Non valide VPCNetwork StateFault
- KMSKeyNotAccessibleFault
- OptionGroupNotFoundFault
- StorageQuotaExceededFault
- DBClusterParameterGroupNotFoundFault

Décrire les DBCluster instantanés (action)

Le nom de la AWS CLI pour cette API est :`describe-db-cluster-snapshots`.

Renvoie des informations sur des instantanés de cluster de bases de données. Cette action d'API prend en charge la pagination.

Demande

- DBClusterIdentifier (dans la CLI : `--db-cluster-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID du cluster de bases de données auprès duquel récupérer la liste des instantanés de cluster de bases de données. Ce paramètre ne peut pas être utilisé avec le paramètre `DBClusterSnapshotIdentifier`. Ce paramètre n'est pas sensible à la casse.

  Contraintes :

  - S'il est fourni, il doit correspondre à l'identifiant d'un existant DBCluster.
- DBClusterSnapshotIdentifier (dans la CLI : `--db-cluster-snapshot-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant d'instantané de cluster de bases de données spécifique à décrire. Ce paramètre ne peut pas être utilisé avec le paramètre `DBClusterIdentifier`. Cette valeur est stockée sous la forme d'une chaîne en minuscules.

  Contraintes :

  - S'il est fourni, il doit correspondre à l'identifiant d'un DBCluster instantané existant.
  - Si cet identifiant est destinée à un instantané automatisé, le paramètre `SnapshotType` doit également être spécifié.
- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Ce paramètre n'est actuellement pas pris en charge.
- IncludePublic (dans la CLI : `--include-public`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true pour inclure les instantanés de cluster de bases de données manuels publics et qui peuvent être copiés ou restaurés par tous les comptes Amazon, et false dans le cas contraire. L'argument par défaut est `false`. La valeur par défaut est false.

  Vous pouvez partager un instantané de cluster de bases de données manuel comme public à l'aide de l'action d'API the section called "Modifier DBCluster SnapshotAttribute".

- IncludeShared (dans la CLI : `--include-shared`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true pour inclure les instantanés de cluster de bases de données manuels partagés à partir d'autres comptes Amazon attestant que ce compte Amazon a reçu l'autorisation de copier ou restaurer, et false dans le cas contraire. L'argument par défaut est `false`.

  Vous pouvez autoriser un compte Amazon à restaurer un instantané de cluster de bases de données manuel à partir d'un autre compte Amazon par l'action d'API the section called "Modifier DBCluster SnapshotAttribute".

- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande `DescribeDBClusterSnapshots` précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.

- SnapshotType (dans la CLI : `--snapshot-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type d'instantanés de cluster de bases de données à renvoyer. Vous pouvez spécifier l'une des valeurs suivantes :

  - `automated` : renvoie tous les instantanés de cluster de bases de données automatiquement créés par Amazon Neptune pour mon compte Amazon.
  - `manual` : renvoie tous les instantanés de cluster de bases de données créés par mon compte Amazon.
  - `shared` : renvoie tous les instantanés de cluster de bases de données manuels partagés avec mon compte AWS.

- `public` - Renvoie tous les instantanés de cluster de bases de données marqués comme publics.

Si vous ne spécifiez pas de valeur `SnapshotType`, les deux instantanés de cluster de bases de données manuels et automatiques sont renvoyés. Vous pouvez inclure des instantanés de cluster de bases de données partagés avec ces résultats en définissant le paramètre `IncludeShared` sur `true`. Vous pouvez inclure des instantanés de cluster de bases de données publics avec ces résultats en définissant le paramètre `IncludePublic` sur `true`.

Les paramètres `IncludeShared` et `IncludePublic` ne s'appliquent pas pour les valeurs `SnapshotType` de `manual` ou `automated`. Le paramètre `IncludePublic` ne s'applique pas lorsque `SnapshotType` est défini sur `shared`. Le paramètre `IncludeShared` ne s'applique pas lorsque `SnapshotType` est défini sur `public`.

Réponse

- DBClusterSnapshots : tableau d'objets DBClusterInstantané.

  Fournit une liste des instantanés de cluster de bases de données pour l'utilisateur.
- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande the section called "Décrire les DBCluster instantanés" précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

Erreurs

- DBClusterSnapshotNotFoundFault

Décrire DBCluster SnapshotAttributes (action)

Le nom de la AWS CLI pour cette API est :`describe-db-cluster-snapshot-attributes`.

Renvoie une liste de noms et de valeurs d'attributs d'instantané de cluster de bases de données pour un instantané de cluster de bases de données manuel.

Lorsque vous partagez des instantanés avec d'autres comptes Amazon, `DescribeDBClusterSnapshotAttributes` renvoie l'`restore`attribut et une liste IDs des

comptes Amazon autorisés à copier ou à restaurer l'instantané manuel du cluster de base de données. Si le paramètre `all` est inclus dans la liste des valeurs pour l'attribut `restore`, l'instantané de cluster de bases de données manuel est public et peut être copié ou restauré par tous les comptes Amazon.

Pour ajouter ou supprimer l'accès d'un compte Amazon afin de copier ou de restaurer un instantané de cluster de bases de données manuel, ou de rendre l'instantané de cluster de bases de données manuel public ou privé, utilisez l'action d'API the section called “Modifier DBCluster SnapshotAttribute”.

Demande

- DBClusterSnapshotIdentifier (dans la CLI : `--db-cluster-snapshot-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de l'instantané de cluster de bases de données pour lequel décrire les attributs.

Réponse

Contient les résultats d'appel réussi à l'action d'API the section called “Décrivez DBCluster SnapshotAttributes”.

Les attributs d'instantané de cluster de bases de données manuel sont utilisés pour autoriser d'autres comptes Amazon à copier ou restaurer un instantané de cluster de bases de données manuel. Pour plus d'informations, consultez l'action d'API the section called “Modifier DBCluster SnapshotAttribute”.

- DBClusterSnapshotAttributes : tableau d'objets DBClusterSnapshotAttribute.

  Liste des attributs et des valeurs de l'instantané de cluster de bases de données manuel.
- DBClusterSnapshotIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de l'instantané de cluster de bases de données manuel sur lequel s'appliquent les attributs.

Erreurs

- DBClusterSnapshotNotFoundFault

Structures :

DBClusterInstantané (structure)

Contient les détails d'un instantané de cluster de bases de données Amazon Neptune.

Ce type de données est utilisé comme élément de réponse dans l'action the section called “Décrire les DBCluster instantanés”.

Champs

- AllocatedStorage : entier de type : `integer` (entier signé de 32 bits).

  Spécifie la taille de stockage allouée en gigaoctets (Gio).
- AvailabilityZones : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la liste des zones de EC2 disponibilité dans lesquelles les instances du snapshot du cluster de base de données peuvent être restaurées.
- ClusterCreateTime— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle le cluster de bases de données a été créé, en heure UTC (Universal Coordinated Time).
- DBClusterIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'identifiant du cluster de bases de données à partir duquel cet instantané de cluster de bases de données a été créé.
- DBClusterSnapshotArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'instantané de cluster de bases de données.
- DBClusterSnapshotIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'identifiant d'un instantané de cluster de bases de données. Doit correspondre à l'identifiant d'un instantané existant.

  Après avoir restauré un cluster de bases de données à l'aide d'un `DBClusterSnapshotIdentifier`, vous devez spécifier le même `DBClusterSnapshotIdentifier` pour toute mise à jour ultérieure de ce cluster. Lorsque vous

spécifiez cette propriété pour une mise à jour, le cluster de bases de données n'est pas restauré de nouveau à partir de l'instantané, et les données de la base de données ne sont pas modifiées.

Toutefois, si vous ne spécifiez pas le `DBClusterSnapshotIdentifier`, un cluster de bases de données vide est créé et le cluster de bases de données d'origine est supprimé. Si vous spécifiez une propriété différente de la propriété précédente de restauration d'instantané, le cluster de bases de données est restauré à partir de l'instantané spécifié par le `DBClusterSnapshotIdentifier`, et le cluster de bases de données d'origine est supprimé.

- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom du moteur de base de données.

- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit la version du moteur de base de données à utiliser pour cet instantané de cluster de bases de données.

- IAMDatabaseAuthenticationEnabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur true (vrai) si le mappage des comptes Amazon Identity and Access Management (IAM) aux comptes de base de données est activé ; sinon, valeur false (faux).

- KmsKeyId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si `StorageEncrypted` a la valeur true (vrai), il s'agit de l'identifiant de clé Amazon KMS pour le cluster de bases de données chiffré.

- LicenseModel : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit les informations sur le modèle de licence pour cet instantané de cluster de bases de données.

- PercentProgress : entier de type : `integer` (entier signé de 32 bits).

  Spécifie une estimation du pourcentage de données transférées.

- Port : entier de type : `integer` (entier signé de 32 bits).

  Spécifie le port écouté par le cluster de bases de données au moment de l'instantané.

- SnapshotCreateTime— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie l'heure à laquelle l'instantané a été pris, au format UTC (temps universel).

- SnapshotType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le type de l'instantané de cluster de bases de données.
- SourceDBClusterSnapshotArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si l'instantané de cluster de bases de données a été copié à partir d'un instantané de cluster de bases de données source, l'Amazon Resource Name (ARN) de l'instantané de cluster de bases de données source, une valeur null dans le cas contraire.
- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le statut de cet instantané de cluster de bases de données.
- StorageEncrypted : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si l'instantané de cluster de bases de données est chiffré.
- StorageType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de stockage associé à l'instantané de cluster de bases de données.
- VpcId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit l'ID de VPC associé à l'instantané de cluster de bases de données.

`DBClusterSnapshot` est utilisé comme élément de réponse pour :

- Créer un DBCluster instantané
- Copier DBCluster un instantané
- Supprimer DBCluster un instantané

DBClusterSnapshotAttribute (structure)

Contient le nom et les valeurs d'un attribut d'instantané de cluster de bases de données manuel.

Les attributs d'instantané de cluster de bases de données manuel sont utilisés pour autoriser d'autres comptes Amazon à restaurer un instantané de cluster de bases de données manuel. Pour plus d'informations, consultez l'action d'API the section called "Modifier DBCluster SnapshotAttribute".

Champs

- AttributeName : chaîne de type : `string` (chaîne encodée en UTF-8).

Nom de l'attribut d'instantané de cluster de bases de données manuel.

L'attribut nommé `restore` fait référence à la liste des comptes Amazon autorisés à copier ou restaurer l'instantané de cluster de bases de données manuel. Pour plus d'informations, consultez l'action d'API the section called “Modifier DBCluster SnapshotAttribute”.

- AttributeValues : chaîne de type : `string` (chaîne encodée en UTF-8).

  Valeur(s) de l'attribut d'instantané de cluster de bases de données manuel.

  Si le `AttributeName` champ est défini sur`restore`, cet élément renvoie une liste IDs des comptes Amazon autorisés à copier ou à restaurer l'instantané manuel du cluster de base de données. Si une valeur `all` se trouve dans la liste, l'instantané de cluster de bases de données manuel est public et disponible pour la copie ou la restauration des comptes Amazon.

DBClusterSnapshotAttributesResult (structure)

Contient les résultats d'appel réussi à l'action d'API the section called “Décrivez DBCluster SnapshotAttributes”.

Les attributs d'instantané de cluster de bases de données manuel sont utilisés pour autoriser d'autres comptes Amazon à copier ou restaurer un instantané de cluster de bases de données manuel. Pour plus d'informations, consultez l'action d'API the section called “Modifier DBCluster SnapshotAttribute”.

Champs

- DBClusterSnapshotAttributes : tableau d'objets DBClusterSnapshotAttribute.

  Liste des attributs et des valeurs de l'instantané de cluster de bases de données manuel.

- DBClusterSnapshotIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de l'instantané de cluster de bases de données manuel sur lequel s'appliquent les attributs.

`DBClusterSnapshotAttributesResult` est utilisé comme élément de réponse pour :

- Décrivez DBCluster SnapshotAttributes
- Modifier DBCluster SnapshotAttribute

API d'événements Neptune

Actions :

- CreateEventSubscription (action)
- DeleteEventSubscription (action)
- ModifyEventSubscription (action)
- DescribeEventSubscriptions (action)
- AddSourceIdentifierToSubscription (action)
- RemoveSourceIdentifierFromSubscription (action)
- DescribeEvents (action)
- DescribeEventCategories (action)

Structures :

- Event (structure)
- EventCategoriesMap (structure)
- EventSubscription (structure)

CreateEventSubscription (action)

> Le nom de la AWS CLI pour cette API est :`create-event-subscription`.

Crée un abonnement à la notification d'événements. Cette action nécessite un ARN (Amazon Resource Name) de rubrique créé par la console Neptune, la console SNS ou l'API SNS. Pour obtenir un ARN avec SNS, vous devez créer une rubrique dans Amazon SNS et vous y abonner. L'ARN s'affiche dans la console SNS.

Vous pouvez spécifier le type de source (SourceType) dont vous souhaitez être informé, fournir une liste des sources Neptune (SourceIds) qui déclenchent les événements et fournir une liste de catégories d'événements (EventCategories) pour les événements dont vous souhaitez être informé. Par exemple, vous pouvez spécifier SourceType = db-instance, = mydbinstance1, mydbinstance2 et SourceIds = Availability, Backup. EventCategories

Si vous spécifiez à la fois le SourceType et SourceIds, par exemple SourceType = db-instance et SourceIdentifier = myDBInstance1, vous êtes informé de tous les événements db-instance pour la

source spécifiée. Si vous spécifiez a SourceType mais pas a SourceIdentifier, vous êtes informé des événements liés à ce type de source pour toutes vos sources Neptune. Si vous ne spécifiez ni le SourceType ni le SourceIdentifier, vous êtes informé des événements générés par toutes les sources Neptune associées à votre compte client.

Demande

- Enabled(dans la CLI :`--enabled`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Valeur booléenne définie sur true pour activer l'abonnement, et définie sur false pour créer l'abonnement sans l'activer.

- EventCategories (dans la CLI : `--event-categories`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des catégories d'événements SourceType auxquels vous souhaitez vous abonner.
  Vous pouvez voir la liste des catégories correspondant à une donnée SourceType en utilisant l'DescribeEventCategoriesaction.

- SnsTopicArn (dans la CLI : `--sns-topic-arn`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de la rubrique SNS créé pour la notification d'événements. L'ARN est créé par Amazon SNS lorsque vous créez une rubrique et vous y abonnez.

- SourceIds (dans la CLI : `--source-ids`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des identifiants des sources d'événements pour lesquels des événements sont renvoyés. Si la valeur n'est pas spécifiée, toutes les sources sont incluses dans la réponse. Un identifiant doit commencer par une lettre et contenir uniquement des lettres ASCII, des chiffres et des tirets. Il ne doit pas se terminer par un tiret ou contenir deux tirets consécutifs.

  Contraintes :

  - S' SourceIds ils sont fournis, ils SourceType doivent également être fournis.
  - Si le type de source est une instance de base de données, un paramètre `DBInstanceIdentifier` doit être fourni.
  - Si le type de source est un groupe de sécurité de base de données, un paramètre `DBSecurityGroupName` doit être fourni.

  - Si le type de source est un groupe de paramètres de base de données, un paramètre `DBParameterGroupName` doit être fourni.
  - Si le type de source est un instantané de base de données, un paramètre `DBSnapshotIdentifier` doit être fourni.
- SourceType (dans la CLI : `--source-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de source qui génère les événements. Par exemple, si vous souhaitez être informé des événements générés par une instance de base de données, vous devez définir ce paramètre sur db-instance. Si cette valeur n'est pas spécifiée, tous les événements sont renvoyés.

  Valeurs valides: `db-instance` | `db-cluster` | `db-parameter-group` | `db-security-group` | `db-snapshot` | `db-cluster-snapshot`
- SubscriptionName (dans la CLI : `--subscription-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de l'abonnement.

  Contraintes : le nom doit être composé de 255 caractères maximum.
- Tags (dans la CLI : `--tags`) : tableau d'objets Tag.

  Balises à appliquer au nouvel abonnement d'événements.

Réponse

Contient les résultats d'une invocation réussie de l'action the section called "DescribeEventSubscriptions".

- CustomerAwsId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Compte client Amazon associé à l'abonnement aux notifications d'événements.
- CustSubscriptionId : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID d'abonnement à la notification d'événements.
- Enabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur booléenne indiquant si l'abonnement est activé. La valeur true indique que l'abonnement est activé.

- EventCategoriesList : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des catégories d'événements pour l'abonnement à la notification d'événements.
- EventSubscriptionArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'abonnement aux événements.
- SnsTopicArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN de la rubrique de l'abonnement à la notification d'événements.
- SourceIdsList : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des sources IDs pour l'abonnement aux notifications d'événements.
- SourceType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de source pour l'abonnement à la notification d'événements.
- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'état de l'abonnement à la notification d'événements .

  Contraintes :

  Il peut s'agir de l'une des options suivantes : création | modification | suppression | active | sans autorisation | topic-not-exist

  Le statut « pas de permission » indique que Neptune n'a plus l'autorisation de publier sur la rubrique SNS. Le statut « topic-not-exist » indique que le sujet a été supprimé après la création de l'abonnement.
- SubscriptionCreationTime : chaîne de type : `string` (chaîne encodée en UTF-8).

  Heure de création de l'abonnement à la notification d'événements.

Erreurs

- EventSubscriptionQuotaExceededFault
- SubscriptionAlreadyExistFault
- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault

- SubscriptionCategoryNotFoundFault
- SourceNotFoundFault

DeleteEventSubscription (action)

Le nom de la AWS CLI pour cette API est :`delete-event-subscription`.

Supprime un abonnement à la notifications d'événements.

Demande

- SubscriptionName (dans la CLI : `--subscription-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de l'abonnement à la notification d'événements que vous souhaitez supprimer.

Réponse

Contient les résultats d'une invocation réussie de l'action the section called "DescribeEventSubscriptions".

- CustomerAwsId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Compte client Amazon associé à l'abonnement aux notifications d'événements.
- CustSubscriptionId : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID d'abonnement à la notification d'événements.
- Enabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur booléenne indiquant si l'abonnement est activé. La valeur true indique que l'abonnement est activé.
- EventCategoriesList : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des catégories d'événements pour l'abonnement à la notification d'événements.
- EventSubscriptionArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'abonnement aux événements.
- SnsTopicArn : chaîne de type : `string` (chaîne encodée en UTF-8).

ARN de la rubrique de l'abonnement à la notification d'événements.

- SourceIdsList : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des sources IDs pour l'abonnement aux notifications d'événements.

- SourceType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de source pour l'abonnement à la notification d'événements.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'état de l'abonnement à la notification d'événements .

  Contraintes :

  Il peut s'agir de l'une des options suivantes : création | modification | suppression | active | sans autorisation | topic-not-exist

  Le statut « pas de permission » indique que Neptune n'a plus l'autorisation de publier sur la rubrique SNS. Le statut « topic-not-exist » indique que le sujet a été supprimé après la création de l'abonnement.

- SubscriptionCreationTime : chaîne de type : `string` (chaîne encodée en UTF-8).

  Heure de création de l'abonnement à la notification d'événements.

Erreurs

- SubscriptionNotFoundFault
- InvalidEventSubscriptionStateFault

ModifyEventSubscription (action)

Le nom de la AWS CLI pour cette API est :`modify-event-subscription`.

Modifie un abonnement à la notification d'événements existant. Notez que vous ne pouvez pas modifier les identifiants source à l'aide de cet appel. Pour modifier les identifiants source d'un abonnement, utilisez les appels commande the section called “AddSourceIdentifierToSubscription” et the section called “RemoveSourceIdentifierFromSubscription”.

Vous pouvez voir la liste des catégories d'événements pour un événement donné SourceType en utilisant l'DescribeEventCategoriesaction.

Demande

- Enabled(dans la CLI :`--enabled`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Valeur booléenne définie sur true pour activer l'abonnement.
- EventCategories (dans la CLI : `--event-categories`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des catégories d'événements SourceType auxquels vous souhaitez vous abonner.
  Vous pouvez voir la liste des catégories correspondant à une donnée SourceType en utilisant l'DescribeEventCategoriesaction.
- SnsTopicArn (dans la CLI : `--sns-topic-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de la rubrique SNS créé pour la notification d'événements. L'ARN est créé par Amazon SNS lorsque vous créez une rubrique et vous y abonnez.
- SourceType (dans la CLI : `--source-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de source qui génère les événements. Par exemple, si vous souhaitez être informé des événements générés par une instance de base de données, vous devez définir ce paramètre sur db-instance. Si cette valeur n'est pas spécifiée, tous les événements sont renvoyés.

  Valeurs valides : db-instance | | db-parameter-group | db-security-group db-snapshot
- SubscriptionName (dans la CLI : `--subscription-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le nom de l'abonnement à la notification d'événements .

Réponse

Contient les résultats d'une invocation réussie de l'action the section called "DescribeEventSubscriptions".

- CustomerAwsId : chaîne de type : `string` (chaîne encodée en UTF-8).

Compte client Amazon associé à l'abonnement aux notifications d'événements.

- CustSubscriptionId : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID d'abonnement à la notification d'événements.

- Enabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur booléenne indiquant si l'abonnement est activé. La valeur true indique que l'abonnement est activé.

- EventCategoriesList : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des catégories d'événements pour l'abonnement à la notification d'événements.

- EventSubscriptionArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'abonnement aux événements.

- SnsTopicArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN de la rubrique de l'abonnement à la notification d'événements.

- SourceIdsList : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des sources IDs pour l'abonnement aux notifications d'événements.

- SourceType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de source pour l'abonnement à la notification d'événements.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'état de l'abonnement à la notification d'événements .

  Contraintes :

  Il peut s'agir de l'une des options suivantes : création | modification | suppression | active | sans autorisation | topic-not-exist

  Le statut « pas de permission » indique que Neptune n'a plus l'autorisation de publier sur la rubrique SNS. Le statut « topic-not-exist » indique que le sujet a été supprimé après la création de l'abonnement.

- SubscriptionCreationTime : chaîne de type : `string` (chaîne encodée en UTF-8).

  Heure de création de l'abonnement à la notification d'événements.

Erreurs

- EventSubscriptionQuotaExceededFault
- SubscriptionNotFoundFault
- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault
- SubscriptionCategoryNotFoundFault

DescribeEventSubscriptions (action)

Le nom de la AWS CLI pour cette API est :`describe-event-subscriptions`.

Répertorie toutes les descriptions d'abonnements d'un compte client. La description d'un abonnement inclut l' SNSTopicARN SubscriptionName, le CustomerID, le SourceID SourceType et le Status. CreationTime

Si vous spécifiez un SubscriptionName, répertorie la description de cet abonnement.

Demande

- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Ce paramètre n'est actuellement pas pris en charge.
- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Un jeton de pagination facultatif fourni par une demande d' DescribeOrderableDBInstanceoptions précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.
- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.

- SubscriptionName (dans la CLI : `--subscription-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de l'abonnement à la notification d'événements que vous souhaitez décrire.

Réponse

- EventSubscriptionsList : tableau d'objets EventSubscription.

  Liste des types de EventSubscriptions données.
- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Un jeton de pagination facultatif fourni par une demande d' DescribeOrderableDBInstanceoptions précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

Erreurs

- SubscriptionNotFoundFault

AddSourceIdentifierToSubscription (action)

Le nom de la AWS CLI pour cette API est :`add-source-identifier-to-subscription`.

Ajoute un identifiant source à un abonnement à la notification d'événements existant.

Demande

- SourceIdentifier (dans la CLI : `--source-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de la source de l'événement à ajouter.

  Contraintes :

  - Si le type de source est une instance de base de données, un paramètre `DBInstanceIdentifier` doit être fourni.
  - Si le type de source est un groupe de sécurité de base de données, un paramètre `DBSecurityGroupName` doit être fourni.

  - Si le type de source est un groupe de paramètres de base de données, un paramètre `DBParameterGroupName` doit être fourni.
  - Si le type de source est un instantané de base de données, un paramètre `DBSnapshotIdentifier` doit être fourni.
- SubscriptionName (dans la CLI : `--subscription-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de l'abonnement à la notification d'événements auquel vous souhaitez ajouter un identifiant source.

Réponse

Contient les résultats d'une invocation réussie de l'action the section called "DescribeEventSubscriptions".

- CustomerAwsId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Compte client Amazon associé à l'abonnement aux notifications d'événements.
- CustSubscriptionId : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID d'abonnement à la notification d'événements.
- Enabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur booléenne indiquant si l'abonnement est activé. La valeur true indique que l'abonnement est activé.
- EventCategoriesList : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des catégories d'événements pour l'abonnement à la notification d'événements.
- EventSubscriptionArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'abonnement aux événements.
- SnsTopicArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN de la rubrique de l'abonnement à la notification d'événements.
- SourceIdsList : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des sources IDs pour l'abonnement aux notifications d'événements.
- SourceType : chaîne de type : `string` (chaîne encodée en UTF-8).

Type de source pour l'abonnement à la notification d'événements.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'état de l'abonnement à la notification d'événements .

  Contraintes :

  Il peut s'agir de l'une des options suivantes : création | modification | suppression | active | sans autorisation | topic-not-exist

  Le statut « pas de permission » indique que Neptune n'a plus l'autorisation de publier sur la rubrique SNS. Le statut « topic-not-exist » indique que le sujet a été supprimé après la création de l'abonnement.
- SubscriptionCreationTime : chaîne de type : `string` (chaîne encodée en UTF-8).

  Heure de création de l'abonnement à la notification d'événements.

Erreurs

- SubscriptionNotFoundFault
- SourceNotFoundFault

RemoveSourceIdentifierFromSubscription (action)

Le nom de la AWS CLI pour cette API est :`remove-source-identifier-from-subscription`.

Supprime un identifiant source d'un abonnement à la notification d'événements existant

Demande

- SourceIdentifier (dans la CLI : `--source-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant source à supprimer de l'abonnement, tel que l'identifiant d'instance DB pour une instance de base de données ou le nom d'un groupe de sécurité.
- SubscriptionName (dans la CLI : `--subscription-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

Nom de l'abonnement à la notification d'événements duquel vous souhaitez supprimer un identifiant source.

Réponse

Contient les résultats d'une invocation réussie de l'action the section called "DescribeEventSubscriptions".

- CustomerAwsId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Compte client Amazon associé à l'abonnement aux notifications d'événements.
- CustSubscriptionId : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID d'abonnement à la notification d'événements.
- Enabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur booléenne indiquant si l'abonnement est activé. La valeur true indique que l'abonnement est activé.
- EventCategoriesList : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des catégories d'événements pour l'abonnement à la notification d'événements.
- EventSubscriptionArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'abonnement aux événements.
- SnsTopicArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN de la rubrique de l'abonnement à la notification d'événements.
- SourceIdsList : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des sources IDs pour l'abonnement aux notifications d'événements.
- SourceType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de source pour l'abonnement à la notification d'événements.
- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'état de l'abonnement à la notification d'événements .

  Contraintes :

Il peut s'agir de l'une des options suivantes : création | modification | suppression | active | sans autorisation | topic-not-exist

Le statut « pas de permission » indique que Neptune n'a plus l'autorisation de publier sur la rubrique SNS. Le statut « topic-not-exist » indique que le sujet a été supprimé après la création de l'abonnement.

- SubscriptionCreationTime : chaîne de type : `string` (chaîne encodée en UTF-8).

  Heure de création de l'abonnement à la notification d'événements.

Erreurs

- SubscriptionNotFoundFault
- SourceNotFoundFault

DescribeEvents (action)

Le nom de la AWS CLI pour cette API est :`describe-events`.

Renvoie des événements associés aux instances de base de données, aux Security Groups DB, aux instantanés de base de données et aux groupes de paramètres de base de données des 14 derniers jours. Des événements propres à une instance de base de données, un Security Group DB, un instantané de base de données ou un groupe de paramètres de base de données en particulier peuvent être obtenus en fournissant le nom en tant que paramètre. Par défaut, la dernière heure d'événements est renvoyée.

Demande

- Duration(dans la CLI :`--duration`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre de minutes pour lesquelles récupérer les événements.

  Par défaut : 60
- EndTime(dans la CLI :`--end-time`) — a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

Fin de l'intervalle de temps pour lequel récupérer les événements au format ISO 8601. Pour plus d'informations sur la norme ISO 8601, rendez-vous sur la page Wikipédia ISO86 01.

Exemple : 2009-07-08T18:00Z

- EventCategories (dans la CLI : `--event-categories`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des catégories d'événements qui déclenchent des notifications pour un abonnement à la notification d'événements.

- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Ce paramètre n'est actuellement pas pris en charge.

- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Un jeton de pagination facultatif fourni par une DescribeEvents demande précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.

- SourceIdentifier (dans la CLI : `--source-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de la source de l'événement pour laquelle les événements sont renvoyés. Si la valeur n'est pas spécifiée, toutes les sources sont incluses dans la réponse.

  Contraintes :

  - S'il SourceIdentifier est fourni, SourceType il doit également être fourni.
  - Si le type de source est `DBInstance`, un paramètre `DBInstanceIdentifier` doit être fourni.

  - Si le type de source est `DBSecurityGroup`, un paramètre `DBSecurityGroupName` doit être fourni.
  - Si le type de source est `DBParameterGroup`, un paramètre `DBParameterGroupName` doit être fourni.
  - Si le type de source est `DBSnapshot`, un paramètre `DBSnapshotIdentifier` doit être fourni.
  - Ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs.
- SourceType(dans la CLI :`--source-type`) — a SourceType, de type : `string` (une chaîne codée en UTF-8).

  Source de l'événement pour laquelle récupérer les événements. Si aucune valeur n'est spécifiée, tous les événements sont renvoyés.
- StartTime(dans la CLI :`--start-time`) — a TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Début de l'intervalle de temps pour lequel récupérer les événements au format ISO 8601. Pour plus d'informations sur la norme ISO 8601, rendez-vous sur la page Wikipédia ISO86 01.

  Exemple : 2009-07-08T18:00Z

Réponse

- Events : tableau d'objets Événement.

  Liste des instances the section called “Événement”.
- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande Events précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

DescribeEventCategories (action)

> Le nom de la AWS CLI pour cette API est :`describe-event-categories`.

Affiche une liste des catégories de tous les types de sources de l'événement ou, si la valeur est spécifiée, d'un type de source donné.

Demande

- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Ce paramètre n'est actuellement pas pris en charge.

- SourceType (dans la CLI : `--source-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de source qui génère les événements.

  Valeurs valides : db-instance | | db-parameter-group | db-security-group db-snapshot

Réponse

- EventCategoriesMapList : tableau d'objets EventCategoriesMap.

  Liste des types de EventCategoriesMap données.

Structures :

Event (structure)

Ce type de données est utilisé comme élément de réponse dans l'action the section called "DescribeEvents".

Champs

- Date— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Spécifie la date et l'heure de l'événement.

- EventCategories : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie la catégorie pour l'événement.

- Message : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit le texte de cet événement.

- SourceArn : chaîne de type : `string` (chaîne encodée en UTF-8).

ARN (Amazon Resource Name) de l'événement.

- SourceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fournit l'identifiant de la source de l'événement.

- SourceType— Il s'agit d'une SourceType, de type : `string` (une chaîne codée en UTF-8).

  Spécifie le type de source pour cet événement.

EventCategoriesMap (structure)

Contient les résultats d'une invocation réussie de l'action the section called "DescribeEventCategories".

Champs

- EventCategories : chaîne de type : `string` (chaîne encodée en UTF-8).

  Catégories d'événements pour le type de source spécifié

- SourceType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de source auquel appartiennent les catégories retournés

EventSubscription (structure)

Contient les résultats d'une invocation réussie de l'action the section called "DescribeEventSubscriptions".

Champs

- CustomerAwsId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Compte client Amazon associé à l'abonnement aux notifications d'événements.

- CustSubscriptionId : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID d'abonnement à la notification d'événements.

- Enabled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

Valeur booléenne indiquant si l'abonnement est activé. La valeur true indique que l'abonnement est activé.

- EventCategoriesList : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des catégories d'événements pour l'abonnement à la notification d'événements.

- EventSubscriptionArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  Amazon Resource Name (ARN) de l'abonnement aux événements.

- SnsTopicArn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN de la rubrique de l'abonnement à la notification d'événements.

- SourceIdsList : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des sources IDs pour l'abonnement aux notifications d'événements.

- SourceType : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de source pour l'abonnement à la notification d'événements.

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'état de l'abonnement à la notification d'événements .

  Contraintes :

  Il peut s'agir de l'une des options suivantes : création | modification | suppression | active | sans autorisation | topic-not-exist

  Le statut « pas de permission » indique que Neptune n'a plus l'autorisation de publier sur la rubrique SNS. Le statut « topic-not-exist » indique que le sujet a été supprimé après la création de l'abonnement.

- SubscriptionCreationTime : chaîne de type : `string` (chaîne encodée en UTF-8).

  Heure de création de l'abonnement à la notification d'événements.

`EventSubscription` est utilisé comme élément de réponse pour :

- CreateEventSubscription
- ModifyEventSubscription

- AddSourceIdentifierToSubscription
- RemoveSourceIdentifierFromSubscription
- DeleteEventSubscription

Autre Neptune APIs

Actions :

- AddTagsToResource (action)
- ListTagsForResource (action)
- RemoveTagsFromResource (action)
- ApplyPendingMaintenanceAction (action)
- DescribePendingMaintenanceActions (action)
- Décrire DBEngine les versions (action)

Structures :

- DBEngineVersion (structure)
- EngineDefaults (structure)
- PendingMaintenanceAction (structure)
- ResourcePendingMaintenanceActions (structure)
- UpgradeTarget (structure)
- Tag (structure)

AddTagsToResource (action)

> Le nom de la AWS CLI pour cette API est :`add-tags-to-resource`.

Ajoute des balises de métadonnées à une ressource Amazon Neptune. Ces balises peuvent être utilisées avec des rapports de répartition des coûts pour suivre les coûts associés aux ressources Amazon Neptune, ou dans une instruction de condition d'une stratégie IAM pour Amazon Neptune.

Demande

- ResourceName (dans la CLI : `--resource-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Ressource Amazon Neptune à laquelle les balises sont ajoutées. Cette valeur est un Amazon Resource Name (ARN). Pour plus d'informations sur la création d'un ARN, consultez Création d'un Amazon Resource Name (ARN).
- Tags (dans la CLI : `--tags`) : obligatoire : tableau d'objets Tag.

  Balises à attribuer à la ressource Amazon Neptune.

Réponse

- Paramètres d'absence de réponse.

Erreurs

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

ListTagsForResource (action)

Le nom de la AWS CLI pour cette API est :`list-tags-for-resource`.

Répertorie toutes les balises sur une ressource Amazon Neptune.

Demande

- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Ce paramètre n'est actuellement pas pris en charge.
- ResourceName (dans la CLI : `--resource-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Ressource Amazon Neptune avec balises à répertorier. Cette valeur est un Amazon Resource Name (ARN). Pour plus d'informations sur la création d'un ARN, consultez Création d'un Amazon Resource Name (ARN).

Réponse

- TagList : tableau d'objets Tag.

  Liste des balises renvoyées par l' ListTagsForResource opération.

Erreurs

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

RemoveTagsFromResource (action)

Le nom de la AWS CLI pour cette API est :`remove-tags-from-resource`.

Supprime des balises de métadonnées d'une ressource Amazon Neptune.

Demande

- ResourceName (dans la CLI : `--resource-name`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Ressource Amazon Neptune de laquelle les balises sont supprimées. Cette valeur est un Amazon Resource Name (ARN). Pour plus d'informations sur la création d'un ARN, consultez Création d'un Amazon Resource Name (ARN).
- TagKeys (dans la CLI : `--tag-keys`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Clé de balise (nom) de la balise à supprimer.

Réponse

- Paramètres d'absence de réponse.

Erreurs

- DBInstanceNotFoundFault

- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

ApplyPendingMaintenanceAction (action)

Le nom de la AWS CLI pour cette API est :`apply-pending-maintenance-action`.

Applique une action de maintenance en attente à une ressource (par exemple, à une instance de base de données).

Demande

- ApplyAction (dans la CLI : `--apply-action`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Action de maintenance en attente à appliquer à cette ressource.

  Valeurs valides : `system-update`, `db-upgrade`
- OptInType (dans la CLI : `--opt-in-type`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Valeur qui spécifie le type de demande de confirmation de l'acceptation ou annule cette dernière. Un type demande de confirmation de l'acceptation de type `immediate` ne peut pas être annulée.

  Valeurs valides :
  - `immediate` - Appliquer immédiatement l'action de maintenance.
  - `next-maintenance` - Appliquer l'action de maintenance pendant le créneau de maintenance suivant pour la ressource.
  - `undo-opt-in` - Annuler toute demande de confirmation de l'acceptation `next-maintenance` existante.
- ResourceIdentifier (dans la CLI : `--resource-identifier`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN (Amazon Resource Name) de la ressource à laquelle s'applique l'action de maintenance en attente. Pour plus d'informations sur la création d'un ARN, consultez Création d'un Amazon Resource Name (ARN).

Réponse

Décrit les actions de maintenance en attente pour une ressource.

- PendingMaintenanceActionDetails : tableau d'objets PendingMaintenanceAction.

  Liste qui fournit des détails sur les actions de maintenance en attente pour la ressource.

- ResourceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN de la ressource qui possède les actions de maintenance en attente.

Erreurs

- ResourceNotFoundFault

DescribePendingMaintenanceActions (action)

Le nom de la AWS CLI pour cette API est :`describe-pending-maintenance-actions`.

Renvoie une liste des ressources (par exemple, des instances de base de données) ayant au moins une action de maintenance en attente.

Demande

- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Filtre qui spécifie une ou plusieurs ressources pour lesquelles renvoyer des actions de maintenance en attente.

  Filtres pris en charge :

  - `db-cluster-id`- Accepte les identifiants de cluster de base de données et les Amazon Resource Names (ARNs) du cluster de bases de données. La liste des résultats inclura uniquement les actions de maintenance en attente pour les clusters de base de données identifiés par ceux-ci ARNs.
  - `db-instance-id`- Accepte les identifiants d'instance de base de données et les instances ARNs de base de données. La liste des résultats inclura uniquement les actions de maintenance en attente pour les instances de base de données identifiées par celles-ci ARNs.

- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

Jeton de pagination facultatif fourni par une demande `DescribePendingMaintenanceActions` précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à un nombre d'enregistrements spécifié par `MaxRecords`.

- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords` spécifiée, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats restants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.
- ResourceIdentifier (dans la CLI : `--resource-identifier`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN d'une ressource pour laquelle renvoyer des actions de maintenance en attente.

Réponse

- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande `DescribePendingMaintenanceActions` précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à un nombre d'enregistrements spécifié par `MaxRecords`.
- PendingMaintenanceActions : tableau d'objets ResourcePendingMaintenanceActions.

  Liste des actions de maintenance en attente pour la ressource.

Erreurs

- ResourceNotFoundFault

Décrire DBEngine les versions (action)

Le nom de la AWS CLI pour cette API est :`describe-db-engine-versions`.

Renvoie une liste des moteurs de base de données disponibles.

Demande

- DBParameterGroupFamily (dans la CLI : `--db-parameter-group-family`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom d'une famille de groupe de paramètres de base de données spécifique pour laquelle renvoyer les détails.

  Contraintes :

  - S'il est fourni, il doit correspondre à un existant DBParameterGroupFamily.
- DefaultOnly (dans la CLI : `--default-only`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique que seule la version par défaut du moteur spécifié ou de l'association moteur et version majeure est renvoyée.
- Engine (dans la CLI : `--engine`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Moteur de base de données à renvoyer.
- EngineVersion (dans la CLI : `--engine-version`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Version du moteur de base de données à renvoyer.

  Exemple : `5.1.49`
- Filters (dans la CLI : `--filters`) : tableau d'objets Filtre.

  Cette option n'est pas prise en charge actuellement.
- ListSupportedCharacterSets(dans la CLI :`--list-supported-character-sets`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

  Si ce paramètre est spécifié et que le moteur demandé prend en charge le paramètre `CharacterSetName` pour `CreateDBInstance`, la réponse inclut une liste des jeux de caractères pris en charge pour chaque version de moteur.
- ListSupportedTimezones(dans la CLI :`--list-supported-timezones`) — a BooleanOptional, de type : `boolean` (une valeur booléenne (vraie ou fausse)).

Si ce paramètre est spécifié et que le moteur demandé prend en charge le paramètre `TimeZone` pour `CreateDBInstance`, la réponse inclut une liste des fuseaux horaires pris en charge pour chaque version de moteur.

- Marker (dans la CLI : `--marker`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

- MaxRecords(dans la CLI :`--max-records`) — un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Nombre maximal d'enregistrements à inclure dans la réponse. Si le nombre d'enregistrements existants est supérieur à la valeur `MaxRecords`, un jeton de pagination appelé marqueur est inclus dans la réponse pour permettre la récupération des résultats suivants.

  Par défaut : 100

  Contraintes : Minimum 20, maximum 100.

Réponse

- DBEngineVersions : tableau d'objets DBEngineVersion.

  Une liste des éléments `DBEngineVersion`.

- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de pagination facultatif fourni par une demande précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.

Structures :

DBEngineVersion (structure)

Ce type de données est utilisé comme élément de réponse dans l'action the section called “Décrire DBEngine les versions”.

Champs

- DBEngineDescription : chaîne de type : `string` (chaîne encodée en UTF-8).

  Description du moteur de base de données.
- DBEngineVersionDescription : chaîne de type : `string` (chaîne encodée en UTF-8).

  Description de la version du moteur de base de données.
- DBParameterGroupFamily : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la famille de groupe de paramètres de base de données pour le moteur de base de données.
- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du moteur de base de données.
- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le numéro de version du moteur de base de données.
- ExportableLogTypes : chaîne de type : `string` (chaîne encodée en UTF-8).

  Types de journaux disponibles pour exportation vers CloudWatch Logs par le moteur de base de données.
- SupportedTimezones : tableau d'objets Fuseau horaire.

  Liste des fuseaux horaires pris en charge par ce moteur pour le paramètre `Timezone` de l'action `CreateDBInstance`.
- SupportsGlobalDatabases : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur qui indique si vous pouvez utiliser les bases de données globales Neptune avec une version de moteur de base de données spécifique.
- SupportsLogExportsToCloudwatchLogs : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Une valeur qui indique si la version du moteur prend en charge l'exportation des types de journaux spécifiés par ExportableLogTypes to CloudWatch Logs.
- SupportsReadReplica : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si la version du moteur de base de données prend en charge les réplicas en lecture.

- ValidUpgradeTarget : tableau d'objets UpgradeTarget.

  Liste des versions de moteur vers lesquelles cette version du moteur de base de données peut être mise à niveau.

EngineDefaults (structure)

Contient le résultat d'une invocation réussie de l'action the section called "DescribeEngineDefaultParameters".

Champs

- DBParameterGroupFamily : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie le nom de la famille de groupe de paramètres de base de données à laquelle s'appliquent les paramètres par défaut du moteur.
- Marker : chaîne de type : `string` (chaîne encodée en UTF-8).

  Un jeton de pagination facultatif fourni par une EngineDefaults demande précédente. Si ce paramètre est spécifié, la réponse inclut uniquement des enregistrements supérieurs au marqueur, jusqu'à la valeur spécifiée par `MaxRecords`.
- Parameters : tableau d'objets Paramètre.

  Contient une liste des paramètres par défaut du moteur.

`EngineDefaults` est utilisé comme élément de réponse pour :

- DescribeEngineDefaultParameters
- DescribeEngineDefaultClusterParameters

PendingMaintenanceAction (structure)

Fournit des informations sur une action de maintenance en attente pour une ressource.

Champs

- Action : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type d'action de maintenance en attente disponible pour la ressource.

- AutoAppliedAfterDate— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Date de la fenêtre de maintenance lorsque l'action est appliquée. L'action de maintenance est appliquée à la ressource lors de sa première fenêtre de maintenance après cette date. Si cette date est spécifiée, toutes les demandes de confirmation de l'acceptation `next-maintenance` sont ignorées.

- CurrentApplyDate— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Date effective d'application de l'action de maintenance en attente à la ressource. Cette date tient compte des demandes de confirmation de l'acceptation de compte reçues à partir de l'API the section called "ApplyPendingMaintenanceAction", le paramètre `AutoAppliedAfterDate` et le paramètre `ForcedApplyDate`. Ce champ est vide si une demande de confirmation de l'acceptation n'a pas été reçue et rien n'a été spécifié comme `AutoAppliedAfterDate` ou `ForcedApplyDate`.

- Description : chaîne de type : `string` (chaîne encodée en UTF-8).

  Description fournissant plus de détails sur l'action de maintenance.

- ForcedApplyDate— Il s'agit d'un TStamp, de type : `timestamp` (un point dans le temps, généralement défini comme un décalage par rapport à minuit le 01/01/1970).

  Date à laquelle l'action de maintenance est automatiquement appliquée. L'action de maintenance est appliquée à la ressource à cette date indépendamment de la fenêtre de maintenance de la ressource. Si cette date est spécifiée, toutes les demandes de confirmation de l'acceptation `immediate` sont ignorées.

- OptInStatus : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique le type de demande de confirmation de l'acceptation reçue pour la ressource.

ResourcePendingMaintenanceActions (structure)

Décrit les actions de maintenance en attente pour une ressource.

Champs

- PendingMaintenanceActionDetails : tableau d'objets PendingMaintenanceAction.

Liste qui fournit des détails sur les actions de maintenance en attente pour la ressource.

- ResourceIdentifier : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN de la ressource qui possède les actions de maintenance en attente.

`ResourcePendingMaintenanceActions` est utilisé comme élément de réponse pour :

- ApplyPendingMaintenanceAction

UpgradeTarget (structure)

Version du moteur de base de données vers laquelle une instance de base de données peut être mise à niveau.

Champs

- AutoUpgrade : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Une valeur qui indique si la version cible est appliquée à toutes les instances de base de données source AutoMinorVersionUpgrade définies sur true.
- Description : chaîne de type : `string` (chaîne encodée en UTF-8).

  Version du moteur de base de données vers laquelle une instance de base de données peut être mise à niveau.
- Engine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du moteur de base de données cible mis à niveau.
- EngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

  Numéro de version du moteur de base de données cible mis à niveau.
- IsMajorVersionUpgrade : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur qui indique si un moteur de base de données est mis à niveau vers une version majeure.
- SupportsGlobalDatabases— Il s'agit d'une BooleanOptional valeur booléenne `boolean` (vraie ou fausse) du type : (valeur booléenne).

Valeur qui indique si vous pouvez utiliser les bases de données globales Neptune avec la version de moteur cible.

Tag (structure)

Métadonnée attribuée à une ressource Amazon Neptune composée d'une paire clé-valeur.

Champs

- Key : chaîne de type : `string` (chaîne encodée en UTF-8).

  Une clé est le nom obligatoire de la balise. La valeur de la chaîne peut comporter de 1 à 128 caractères Unicode et elle ne peut pas être précédée de la mention `aws:` ou `rds:`. La chaîne peut uniquement contenir l'ensemble de lettres, de chiffres et d'espaces Unicode, '_', '.', '/', '=', '+', '-' (regex Java : "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$").
- Value : chaîne de type : `string` (chaîne encodée en UTF-8).

  Une valeur est la valeur facultative de la balise. La valeur de la chaîne peut comporter de 1 à 256 caractères Unicode et elle ne peut pas être précédée de la mention `aws:` ou `rds:`. La chaîne peut uniquement contenir l'ensemble de lettres, de chiffres et d'espaces Unicode, '_', '.', '/', '=', '+', '-' (regex Java : "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$").

Types de données Neptune courants

Structures :

- AvailabilityZone (structure)
- DBSecurityGroupMembership (structure)
- DomainMembership (structure)
- DoubleRange (structure)
- Endpoint (structure)
- Filter (structure)
- Range (structure)
- ServerlessV2 ScalingConfiguration (structure)
- ServerlessV2 ScalingConfigurationInfo (structure)

- Timezone (structure)
- VpcSecurityGroupMembership (structure)

AvailabilityZone (structure)

Spécifie une zone de disponibilité.

Champs

- Name : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la zone de disponibilité.

DBSecurityGroupMembership (structure)

Spécifie l'appartenance à un Security Group DB désigné.

Champs

- DBSecurityGroupName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du Security Group DB.
- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Statut du Security Group DB.

DomainMembership (structure)

Enregistrement de l'appartenance au domaine Active Directory associé à une instance de base de données.

Champs

- Domain : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant du domaine Active Directory.
- FQDN : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de domaine complet du domaine Active Directory.

- IAMRoleName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du rôle IAM à utiliser pour effectuer des appels d'API à Directory Service.
- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Statut de l'instance de l'appartenance au domaine Active Directory de l'instance de base de données, tel que joint, en attente, échec, etc.).

DoubleRange (structure)

Plage de valeurs doubles.

Champs

- From : double de type : `double` (nombre à virgule flottante IEEE 754 à double précision).

  Valeur minimum de la plage.
- To : double de type : `double` (nombre à virgule flottante IEEE 754 à double précision).

  Valeur maximum de la plage.

Endpoint (structure)

Spécifie un point de terminaison de connexion.

Pour la structure de données qui représente les points de terminaison du cluster de bases de données Amazon Neptune, consultez `DBClusterEndpoint`.

Champs

- Address : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'adresse DNS de l'instance de base de données.
- HostedZoneId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Spécifie l'ID attribué par Amazon Route 53 lorsque vous créez une zone hébergée.
- Port : entier de type : `integer` (entier signé de 32 bits).

  Spécifie le port sur lequel le moteur de base de données est à l'écoute.

Filter (structure)

Ce type n'est actuellement pas pris en charge.

Champs

- Name : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Ce paramètre n'est actuellement pas pris en charge.
- Values : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Ce paramètre n'est actuellement pas pris en charge.

Range (structure)

Plage de valeurs entières.

Champs

- From : entier de type : `integer` (entier signé de 32 bits).

  Valeur minimum de la plage.
- Step— Il s'agit d'un IntegerOptional, de type : `integer` (un entier signé de 32 bits).

  Valeur de pas de la plage. Par exemple, si vous avez une plage de 5 000 à 10 000, avec une valeur de pas de 1 000, les valeurs valides commencent à 5 000 et augmentent de 1 000. Même si 7 500 se trouve dans la plage, ce n'est pas une valeur valide. Les valeurs valides sont 5 000, 6 000, 7 000 et 8 000...
- To : entier de type : `integer` (entier signé de 32 bits).

  Valeur maximum de la plage.

ServerlessV2 ScalingConfiguration (structure)

Contient la configuration de mise à l'échelle d'un cluster de bases de données Neptune sans serveur.

Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

Champs

- MaxCapacity— Il s'agit d'un DoubleOptional, de type : `double` (un nombre à virgule flottante IEEE 754 à double précision).

  Nombre maximal d'unités de capacité Neptune (NCUs) pour une instance de base de données dans un cluster Neptune Serverless. Vous pouvez spécifier les valeurs NCU par paliers de 0,5, par exemple 40, 40,5, 41, etc.
- MinCapacity— Il s'agit d'un DoubleOptional, de type : `double` (un nombre à virgule flottante IEEE 754 à double précision).

  Le nombre minimum d'unités de capacité Neptune (NCUs) pour une instance de base de données dans un cluster Neptune Serverless. Vous pouvez spécifier les valeurs NCU par paliers de 0,5, par exemple 8, 8,5, 9, etc.

ServerlessV2 ScalingConfigurationInfo (structure)

Affiche la configuration de mise à l'échelle pour un cluster de bases de données Neptune sans serveur.

Pour plus d'informations, consultez Utilisation d'Amazon Neptune sans serveur dans le Guide de l'utilisateur Amazon Neptune.

Champs

- MaxCapacity— Il s'agit d'un DoubleOptional, de type : `double` (un nombre à virgule flottante IEEE 754 à double précision).

  Nombre maximal d'unités de capacité Neptune (NCUs) pour une instance de base de données dans un cluster Neptune Serverless. Vous pouvez spécifier les valeurs NCU par paliers de 0,5, par exemple 40, 40,5, 41, etc.
- MinCapacity— Il s'agit d'un DoubleOptional, de type : `double` (un nombre à virgule flottante IEEE 754 à double précision).

  Le nombre minimum d'unités de capacité Neptune (NCUs) pour une instance de base de données dans un cluster Neptune Serverless. Vous pouvez spécifier les valeurs NCU par paliers de 0,5, par exemple 8, 8,5, 9, etc.

Timezone (structure)

Un fuseau horaire associé à une the section called "DBInstance".

Champs

- TimezoneName : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du fuseau horaire.

VpcSecurityGroupMembership (structure)

Type de données utilisé comme un élément de réponse pour les requêtes sur l'appartenance de groupe de sécurité VPC.

Champs

- Status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Statut du groupe de sécurité VPC.
- VpcSecurityGroupId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du groupe de sécurité VPC.

Neptune : exceptions spécifiques à un individu APIs

Exceptions :

- AuthorizationAlreadyExistsFault (structure)
- AuthorizationNotFoundFault (structure)
- AuthorizationQuotaExceededFault (structure)
- CertificateNotFoundFault (structure)
- DBClusterAlreadyExistsFault (structure)
- DBClusterNotFoundFault (structure)
- DBClusterParameterGroupNotFoundFault (structure)
- DBClusterQuotaExceededFault (structure)
- DBClusterRoleAlreadyExistsFault (structure)

- DBClusterRoleNotFoundFault (structure)
- DBClusterRoleQuotaExceededFault (structure)
- DBClusterSnapshotAlreadyExistsFault (structure)
- DBClusterSnapshotNotFoundFault (structure)
- DBInstanceAlreadyExistsFault (structure)
- DBInstanceNotFoundFault (structure)
- DBLogFileNotFoundFault (structure)
- DBParameterGroupAlreadyExistsFault (structure)
- DBParameterGroupNotFoundFault (structure)
- DBParameterGroupQuotaExceededFault (structure)
- DBSecurityGroupAlreadyExistsFault (structure)
- DBSecurityGroupNotFoundFault (structure)
- DBSecurityGroupNotSupportedFault (structure)
- DBSecurityGroupQuotaExceededFault (structure)
- DBSnapshotAlreadyExistsFault (structure)
- DBSnapshotNotFoundFault (structure)
- DBSubnetGroupAlreadyExistsFault (structure)
- DBSubnetGroupDoesNotCoverEnoughAZs (structure)
- DBSubnetGroupNotAllowedFault (structure)
- DBSubnetGroupNotFoundFault (structure)
- DBSubnetGroupQuotaExceededFault (structure)
- DBSubnetQuotaExceededFault (structure)
- DBUpgradeDependencyFailureFault (structure)
- DomainNotFoundFault (structure)
- EventSubscriptionQuotaExceededFault (structure)
- GlobalClusterAlreadyExistsFault (structure)
- GlobalClusterNotFoundFault (structure)
- GlobalClusterQuotaExceededFault (structure)
- InstanceQuotaExceededFault (structure)
- Insuffisant DBCluster CapacityFault (structure)

- Insuffisant DBInstance CapacityFault (structure)
- InsufficientStorageClusterCapacityFault (structure)
- Non valide DBCluster EndpointStateFault (structure)
- Non valide DBCluster SnapshotStateFault (structure)
- Non valide DBCluster StateFault (structure)
- Non valide DBInstance StateFault (structure)
- Non valide DBParameter GroupStateFault (structure)
- Non valide DBSecurity GroupStateFault (structure)
- Non valide DBSnapshot StateFault (structure)
- Non valide DBSubnet GroupFault (structure)
- Non valide DBSubnet GroupStateFault (structure)
- Non valide DBSubnet StateFault (structure)
- InvalidEventSubscriptionStateFault (structure)
- InvalidGlobalClusterStateFault (structure)
- InvalidOptionGroupStateFault (structure)
- InvalidRestoreFault (structure)
- InvalidSubnet (structure)
- Non valide VPCNetwork StateFault (structure)
- KMSKeyNotAccessibleFault (structure)
- OptionGroupNotFoundFault (structure)
- PointInTimeRestoreNotEnabledFault (structure)
- ProvisionedIopsNotAvailableInAZFault (structure)
- ResourceNotFoundFault (structure)
- SNSInvalidTopicFault (structure)
- SNSNoAuthorizationFault (structure)
- SNSTopicArnNotFoundFault (structure)
- SharedSnapshotQuotaExceededFault (structure)
- SnapshotQuotaExceededFault (structure)
- SourceNotFoundFault (structure)
- StorageQuotaExceededFault (structure)

- StorageTypeNotSupportedFault (structure)
- SubnetAlreadyInUse (structure)
- SubscriptionAlreadyExistFault (structure)
- SubscriptionCategoryNotFoundFault (structure)
- SubscriptionNotFoundFault (structure)

AuthorizationAlreadyExistsFault (structure)

Code de statut HTTP retourné : 400.

Le CIDRIP ou le groupe de EC2 sécurité spécifié est déjà autorisé pour le groupe de sécurité de base de données spécifié.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

AuthorizationNotFoundFault (structure)

Code de statut HTTP retourné : 404.

Le CIDRIP ou le groupe EC2 de sécurité spécifié n'est pas autorisé pour le groupe de sécurité de base de données spécifié.

Neptune peut également ne pas être autorisé via IAM pour exécuter les actions nécessaires en votre nom.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

AuthorizationQuotaExceededFault (structure)

Code de statut HTTP retourné : 400.

Un quota d'autorisation du Security Group DB a été atteint.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

CertificateNotFoundFault (structure)

Code de statut HTTP retourné : 404.

CertificateIdentifierne fait pas référence à un certificat existant.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBClusterAlreadyExistsFault (structure)

Code de statut HTTP retourné : 400.

Un utilisateur possède déjà un cluster de bases de données avec l'identifiant donné.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBClusterNotFoundFault (structure)

Code de statut HTTP retourné : 404.

DBClusterL'identifiant ne fait pas référence à un cluster de base de données existant.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

Message décrivant les détails du problème.

DBClusterParameterGroupNotFoundFault (structure)

Code de statut HTTP retourné : 404.

DBClusterParameterGroupNamene fait pas référence à un groupe de paramètres de cluster de base de données existant.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBClusterQuotaExceededFault (structure)

Code de statut HTTP retourné : 403.

Un utilisateur a tenté de créer un nouveau cluster de bases de données et il a déjà atteint le quota maximum de cluster de bases de données autorisé.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBClusterRoleAlreadyExistsFault (structure)

Code de statut HTTP retourné : 400.

L'ARN (Amazon Resource Name) du rôle IAM spécifié est déjà associé au cluster de bases de données spécifié.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

Message décrivant les détails du problème.

DBClusterRoleNotFoundFault (structure)

Code de statut HTTP retourné : 404.

L'ARN (Amazon Resource Name) du rôle IAM spécifié n'est pas associé au cluster de bases de données spécifié.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBClusterRoleQuotaExceededFault (structure)

Code de statut HTTP retourné : 400.

Vous avez dépassé le nombre maximal de rôles IAM pouvant être associés au cluster de bases de données spécifié.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBClusterSnapshotAlreadyExistsFault (structure)

Code de statut HTTP retourné : 400.

Un utilisateur possède déjà un instantané de cluster de bases de données avec l'identifiant donné.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBClusterSnapshotNotFoundFault (structure)

Code de statut HTTP retourné : 404.

DBClusterSnapshotIdentifierne fait pas référence à un instantané de cluster de base de données existant.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBInstanceAlreadyExistsFault (structure)

Code de statut HTTP retourné : 400.

Un utilisateur possède déjà une instance de base de données avec l'identifiant donné.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBInstanceNotFoundFault (structure)

Code de statut HTTP retourné : 404.

DBInstanceL'identifiant ne fait pas référence à une instance de base de données existante.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBLogFileNotFoundFault (structure)

Code de statut HTTP retourné : 404.

LogFileNamene fait pas référence à un fichier journal de base de données existant.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBParameterGroupAlreadyExistsFault (structure)

Code de statut HTTP retourné : 400.

Un groupe de paramètres de base de données avec le même nom existe.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBParameterGroupNotFoundFault (structure)

Code de statut HTTP retourné : 404.

DBParameterGroupNamene fait pas référence à un groupe de paramètres de base de données existant.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBParameterGroupQuotaExceededFault (structure)

Code de statut HTTP retourné : 400.

Une demande entraînerait le dépassement par l'utilisateur du nombre de groupes de paramètres de base de données.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBSecurityGroupAlreadyExistsFault (structure)

Code de statut HTTP retourné : 400.

Un groupe de sécurité de base de données dont le nom est spécifié dans existe DBSecurityGroupNamedéjà.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBSecurityGroupNotFoundFault (structure)

Code de statut HTTP retourné : 404.

DBSecurityGroupNamene fait pas référence à un groupe de sécurité de base de données existant.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBSecurityGroupNotSupportedFault (structure)

Code de statut HTTP retourné : 400.

Un Security Group DB n'est pas autorisé pour cette action.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

Message décrivant les détails du problème.

DBSecurityGroupQuotaExceededFault (structure)

Code de statut HTTP retourné : 400.

Une demande entraînerait le dépassement par l'utilisateur du nombre de Security Groups DB.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBSnapshotAlreadyExistsFault (structure)

Code de statut HTTP retourné : 400.

DBSnapshotL'identifiant est déjà utilisé par un instantané existant.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBSnapshotNotFoundFault (structure)

Code de statut HTTP retourné : 404.

DBSnapshotL'identifiant ne fait pas référence à un instantané de base de données existant.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBSubnetGroupAlreadyExistsFault (structure)

Code de statut HTTP retourné : 400.

DBSubnetGroupNameest déjà utilisé par un groupe de sous-réseaux de base de données existant.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBSubnetGroupDoesNotCoverEnoughAZs (structure)

Code de statut HTTP retourné : 400.

Des sous-réseaux du groupe de sous-réseaux de base de données doivent couvrir au moins deux zones de disponibilité, sauf s'il n'y en a qu'une seule.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBSubnetGroupNotAllowedFault (structure)

Code de statut HTTP retourné : 400.

Indique que le DBSubnet groupe ne doit pas être spécifié lors de la création de répliques de lecture situées dans la même région que l'instance source.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBSubnetGroupNotFoundFault (structure)

Code de statut HTTP retourné : 404.

DBSubnetGroupNamene fait pas référence à un groupe de sous-réseaux de base de données existant.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBSubnetGroupQuotaExceededFault (structure)

Code de statut HTTP retourné : 400.

Une demande entraînerait le dépassement par l'utilisateur du nombre de groupes de sous-réseaux de base de données.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBSubnetQuotaExceededFault (structure)

Code de statut HTTP retourné : 400.

Une demande entraînerait le dépassement par l'utilisateur du nombre de sous-réseaux dans un groupe de base de données.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DBUpgradeDependencyFailureFault (structure)

Code de statut HTTP retourné : 400.

La mise à niveau de base de données a échoué, car une ressource dont dépendait la base de données n'a pas pu être modifiée.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

DomainNotFoundFault (structure)

Code de statut HTTP retourné : 404.

Domain ne fait pas référence à un domaine Active Directory existant.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

EventSubscriptionQuotaExceededFault (structure)

Code de statut HTTP retourné : 400.

Vous avez dépassé le nombre d'événements auxquels vous pouvez vous abonner.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

GlobalClusterAlreadyExistsFault (structure)

Code de statut HTTP retourné : 400.

`GlobalClusterIdentifier` existe déjà. Choisissez un nouvel identifiant de base de données global (nom unique) pour créer un cluster de bases de données global.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

Message décrivant les détails du problème.

GlobalClusterNotFoundFault (structure)

Code de statut HTTP retourné : 404.

`GlobalClusterIdentifier` ne fait pas référence à un cluster de bases de données global existant.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

GlobalClusterQuotaExceededFault (structure)

Code de statut HTTP retourné : 400.

Le nombre de clusters de bases de données globaux pour ce compte est déjà au maximum autorisé.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

InstanceQuotaExceededFault (structure)

Code de statut HTTP retourné : 400.

Une demande entraînerait le dépassement par l'utilisateur du nombre d'instances de base de données.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Insuffisant DBCluster CapacityFault (structure)

Code de statut HTTP retourné : 403.

Le cluster de bases de données ne possède pas la capacité suffisante pour l'opération en cours.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Insuffisant DBInstance CapacityFault (structure)

Code de statut HTTP retourné : 400.

La classe d'instance de base de données spécifiée n'est pas disponible dans la zone de disponibilité spécifiée.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

InsufficientStorageClusterCapacityFault (structure)

Code de statut HTTP retourné : 400.

Il n'y a pas suffisamment de stockage disponible pour l'action en cours. Vous pouvez résoudre cette erreur en mettant à jour votre groupe de sous-réseaux, afin d'utiliser différentes zones de disponibilité avec davantage d'espace de stockage disponible.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Non valide DBCluster EndpointStateFault (structure)

Code de statut HTTP retourné : 400.

L'opération demandée ne peut pas être effectuée sur le point de terminaison tant qu'il est dans cet état.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Non valide DBCluster SnapshotStateFault (structure)

Code de statut HTTP retourné : 400.

La valeur fournie n'est pas un état d'instantané de cluster de bases de données valide.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Non valide DBCluster StateFault (structure)

Code de statut HTTP retourné : 400.

Le cluster de bases de données n'est pas dans un état valide.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Non valide DBInstance StateFault (structure)

Code de statut HTTP retourné : 400.

L'instance de base de données spécifiée n'est pas dans l'état available.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Non valide DBParameter GroupStateFault (structure)

Code de statut HTTP retourné : 400.

Le groupe de paramètres de base de données est en cours d'utilisation ou se trouve dans un état non valide. Si vous essayez de supprimer le groupe de paramètres, vous ne pouvez pas le supprimer lorsqu'il est dans cet état.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Non valide DBSecurity GroupStateFault (structure)

Code de statut HTTP retourné : 400.

L'état du Security Group DB n'autorise pas la suppression.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Non valide DBSnapshot StateFault (structure)

Code de statut HTTP retourné : 400.

L'état de l'instantané de base de données n'autorise pas la suppression.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Non valide DBSubnet GroupFault (structure)

Code de statut HTTP retourné : 400.

Indique que le DBSubnet groupe n'appartient pas au même VPC que celui d'une réplique de lecture interrégionale existante de la même instance source.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Non valide DBSubnet GroupStateFault (structure)

Code de statut HTTP retourné : 400.

Le groupe de sous-réseaux de base de données ne peut pas être supprimé car il est en cours d'utilisation.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Non valide DBSubnet StateFault (structure)

Code de statut HTTP retourné : 400.

Le sous-réseaux de base de données n'est pas dans l'état available.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

Message décrivant les détails du problème.

InvalidEventSubscriptionStateFault (structure)

Code de statut HTTP retourné : 400.

L'abonnement aux événements est dans un état non valide.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

InvalidGlobalClusterStateFault (structure)

Code de statut HTTP retourné : 400.

Le cluster global est dans un état non valide et ne peut pas effectuer l'opération demandée.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

InvalidOptionGroupStateFault (structure)

Code de statut HTTP retourné : 400.

Le groupe d'options n'est pas dans l'état available.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

InvalidRestoreFault (structure)

Code de statut HTTP retourné : 400.

Impossible de restaurer d'une sauvegarde vpc à une instance de base de données non vpc.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

InvalidSubnet (structure)

Code de statut HTTP retourné : 400.

Le sous-réseau demandé n'est pas valide ou plusieurs sous-réseaux qui ne sont pas dans un VPC commun ont été demandés.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Non valide VPCNetwork StateFault (structure)

Code de statut HTTP retourné : 400.

Un groupe de sous-réseaux de base de données ne couvre pas toutes les zones de disponibilité après sa création, en raison du changement d'utilisateur.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

KMSKeyNotAccessibleFault (structure)

Code de statut HTTP retourné : 400.

Erreur lors de l'accès à la clé KMS.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

OptionGroupNotFoundFault (structure)

Code de statut HTTP retourné : 404.

Le groupe d'options désigné est introuvable.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

PointInTimeRestoreNotEnabledFault (structure)

Code de statut HTTP retourné : 400.

DBInstanceL'identifiant de source fait référence à une instance de base de données dont la BackupRetentionPeriodvaleur est égale à 0.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

ProvisionedIopsNotAvailableInAZFault (structure)

Code de statut HTTP retourné : 400.

IOPS provisionnés indisponibles dans la zone de disponibilité spécifiée.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

ResourceNotFoundFault (structure)

Code de statut HTTP retourné : 404.

L'ID de ressource spécifiée est introuvable.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

SNSInvalidTopicFault (structure)

Code de statut HTTP retourné : 400.

La rubrique SNS n'est pas valide.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

SNSNoAuthorizationFault (structure)

Code de statut HTTP retourné : 400.

Il n'y a aucune autorisation SNS.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

SNSTopicArnNotFoundFault (structure)

Code de statut HTTP retourné : 404.

L'ARN de la rubrique SNS est introuvable.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

SharedSnapshotQuotaExceededFault (structure)

Code de statut HTTP retourné : 400.

Vous avez dépassé le nombre maximal de comptes avec lesquels vous pouvez partager un instantané de base de données manuel.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

SnapshotQuotaExceededFault (structure)

Code de statut HTTP retourné : 400.

Une demande entraînerait le dépassement par l'utilisateur du nombre d'instantanés de base de données.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

SourceNotFoundFault (structure)

Code de statut HTTP retourné : 404.

La source est introuvable.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

StorageQuotaExceededFault (structure)

Code de statut HTTP retourné : 400.

Une demande entraînerait le dépassement par l'utilisateur du volume autorisé de stockage disponible sur toutes les instances de base de données.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

StorageTypeNotSupportedFault (structure)

Code de statut HTTP retourné : 400.

StorageTypespécifiée ne peut pas être associée à l'instance de base de données.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

SubnetAlreadyInUse (structure)

Code de statut HTTP retourné : 400.

Le sous-réseau de base de données est déjà en cours d'utilisation dans la zone de disponibilité.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

SubscriptionAlreadyExistFault (structure)

Code de statut HTTP retourné : 400.

Cet abonnement existe déjà.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

SubscriptionCategoryNotFoundFault (structure)

Code de statut HTTP retourné : 404.

La catégorie d'abonnement désigné est introuvable.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

SubscriptionNotFoundFault (structure)

Code de statut HTTP retourné : 404.

L'abonnement désigné est introuvable.

Champs

- message— Il s'agit d'un ExceptionMessage, de type : `string` (une chaîne codée en UTF-8).

  Message décrivant les détails du problème.

Référence de l'API de données Amazon Neptune

Ce chapitre décrit les opérations de l'API de données Neptune que vous pouvez utiliser pour charger les données, y accéder et les gérer dans le graphe Neptune.

L'API de données Neptune fournit une prise en charge de kit SDK pour le chargement de données, l'exécution de requêtes, l'obtention d'informations sur vos données et l'exécution d'opérations de machine learning. Elle prend en charge les langages de requête Gremlin et openCypher dans Neptune et est disponible dans tous les langages de kit SDK. Elle signe automatiquement les demandes d'API et simplifie considérablement l'intégration de Neptune dans les applications.

Table des matières

- Moteur de plan de données Neptune, réinitialisation rapide et structure générale APIs
  - GetEngineStatus (action)
  - ExecuteFastReset (action)
  - Structures de fonctionnement du moteur :
  - QueryLanguageVersion (structure)
  - FastResetToken (structure)
- Requête Neptune APIs
  - ExecuteGremlinQuery (action)
  - ExecuteGremlinExplainQuery (action)
  - ExecuteGremlinProfileQuery (action)
  - ListGremlinQueries (action)
  - GetGremlinQueryStatus (action)
  - CancelGremlinQuery (action)
  - Actions de requête openCypher :
  - ExecuteOpenCypherQuery (action)
  - ExecuteOpenCypherExplainQuery (action)
  - ListOpenCypherQueries (action)
  - GetOpenCypherQueryStatus (action)
  - CancelOpenCypherQuery (action)
  - Structures des requêtes :

- QueryEvalStats (structure)
- GremlinQueryStatus (structure)
- GremlinQueryStatusAttributes (structure)

- API de chargeur en bloc du plan de données Neptune
  - StartLoaderJob (action)
  - GetLoaderJobStatus (action)
  - ListLoaderJobs (action)
  - CancelLoaderJob (action)
  - Structure de chargement en bloc :
  - LoaderIdResult (structure)
- API de plan de données des flux Neptune
  - GetPropertygraphStream (action)
  - Structures de données de flux :
  - PropertygraphRecord (structure)
  - PropertygraphData (structure)
- Statistiques du plan de données Neptune et résumé du graphique APIs
  - GetPropertygraphStatistics (action)
  - ManagePropertygraphStatistics (action)
  - DeletePropertygraphStatistics (action)
  - GetPropertygraphSummary (action)
  - Structures des statistiques :
  - Statistics (structure)
  - StatisticsSummary (structure)
  - DeleteStatisticsValueMap (structure)
  - RefreshStatisticsIdMap (structure)
  - NodeStructure (structure)
  - EdgeStructure (structure)
  - SubjectStructure (structure)
  - PropertygraphSummaryValueMap (structure)
  - PropertygraphSummary (structure)

- API de traitement de données Neptune ML
  - Démarrer MLData ProcessingJob (action)
  - Liste MLData ProcessingJobs (action)
  - Obtenir MLData ProcessingJob (action)
  - Annuler MLData ProcessingJob (action)
  - Structures à usage général du ML :
  - MlResourceDefinition (structure)
  - MlConfigDefinition (structure)
- API d'entraînement de modèle Neptune ML
  - Démarrer MLModel TrainingJob (action)
  - Liste MLModel TrainingJobs (action)
  - Obtenir MLModel TrainingJob (action)
  - Annuler MLModel TrainingJob (action)
  - Structures d'entraînement des modèles :
  - CustomModelTrainingParameters (structure)
- API de transformation de modèle Neptune ML
  - Démarrer MLModel TransformJob (action)
  - Liste MLModel TransformJobs (action)
  - Obtenir MLModel TransformJob (action)
  - Annuler MLModel TransformJob (action)
  - Structures de transformation de modèle :
  - CustomModelTransformParameters (structure)
- API de point de terminaison d'inférence Neptune ML
  - Créer MLEndpoint (action)
  - Liste MLEndpoints (action)
  - Obtenir MLEndpoint (action)
  - Supprimer MLEndpoint (action)
- Exceptions relatives à l'API du plan de données Neptune
  - AccessDeniedException (structure)
  - BadRequestException (structure)

- BulkLoadIdNotFoundException (structure)
- CancelledByUserException (structure)
- ClientTimeoutException (structure)
- ConcurrentModificationException (structure)
- ConstraintViolationException (structure)
- ExpiredStreamException (structure)
- FailureByQueryException (structure)
- IllegalArgumentException (structure)
- InternalFailureException (structure)
- InvalidArgumentException (structure)
- InvalidNumericDataException (structure)
- InvalidParameterException (structure)
- LoadUrlAccessDeniedException (structure)
- MalformedQueryException (structure)
- MemoryLimitExceededException (structure)
- MethodNotAllowedException (structure)
- MissingParameterException (structure)
- MLResourceNotFoundException (structure)
- ParsingException (structure)
- PreconditionsFailedException (structure)
- QueryLimitExceededException (structure)
- QueryLimitException (structure)
- QueryTooLargeException (structure)
- ReadOnlyViolationException (structure)
- S3Exception (structure)
- ServerShutdownException (structure)
- StatisticsNotAvailableException (structure)
- StreamRecordsNotFoundException (structure)
- ThrottlingException (structure)
- TimeLimitExceededException (structure)

- TooManyRequestsException (structure)
- UnsupportedOperationException (structure)
- UnloadUrlAccessDeniedException (structure)

Moteur de plan de données Neptune, réinitialisation rapide et structure générale APIs

Fonctionnement du moteur :

- GetEngineStatus (action)
- ExecuteFastReset (action)

Structures de fonctionnement du moteur :

- QueryLanguageVersion (structure)
- FastResetToken (structure)

GetEngineStatus (action)

> Le nom de la AWS CLI pour cette API est :`get-engine-status`.

Récupère le statut de la base de données orientée graphe sur l'hôte.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action GetEngineStatusneptune-db : IAM dans ce cluster.

Demande

- Aucun paramètre de demande.

Réponse

- dbEngineVersion : chaîne de type : `string` (chaîne encodée en UTF-8).

Définissez la version de moteur Neptune exécutée sur votre cluster de bases de données. Si cette version de moteur a été corrigée manuellement depuis sa publication, le numéro de version est préfixé par `Patch-`.

- dfeQueryEngine : chaîne de type : `string` (chaîne encodée en UTF-8).

  Définie sur `enabled` si le moteur DFE est complètement activé, ou sur `viaQueryHint` (valeur par défaut) si le moteur DFE n'est utilisé qu'avec les requêtes dont l'indicateur de requête `useDFE` est défini sur. `true`
- features : tableau de mappage de paires clé-valeur où :

  Chaque clé est une chaîne de type : `string` (chaîne encodée en UTF-8).

  Chaque valeur est un document de type : `document` (contenu ouvert indépendant du protocole représenté par un modèle de données de type JSON).

  Contient des informations d'état sur les fonctionnalités activées sur votre cluster de bases de données.
- gremlin : objet QueryLanguageVersion.

  Contient des informations sur le langage de requête Gremlin disponible sur votre cluster. Plus précisément, il contient un champ de version qui indique la TinkerPop version actuelle utilisée par le moteur.
- labMode : tableau de mappage de paires clé-valeur où :

  Chaque clé est une chaîne de type : `string` (chaîne encodée en UTF-8).

  Chaque valeur est une chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient les paramètres de mode expérimental utilisés par le moteur.
- opencypher : objet QueryLanguageVersion.

  Contient des informations sur le langage de requête openCypher disponible sur votre cluster. Plus précisément, il contient un champ de version qui spécifie la version actuelle d'openCypher utilisée par le moteur.
- role : chaîne de type : `string` (chaîne encodée en UTF-8).

  Définie sur `reader` si l'instance est un réplica en lecture ou sur `writer` si l'instance est l'instance principale.

- rollingBackTrxCount : entier de type : `integer` (entier signé de 32 bits).

  Si des transactions sont annulées, ce champ est défini sur le nombre de transactions de ce type. S'il n'y en a pas, ce champ ne s'affiche pas.

- rollingBackTrxEarliestStartTime : chaîne de type : `string` (chaîne encodée en UTF-8).

  Heure de début de la transaction la plus ancienne annulée. Si aucune transaction n'est annulée, ce champ ne s'affiche pas.

- settings : tableau de mappage de paires clé-valeur où :

  Chaque clé est une chaîne de type : `string` (chaîne encodée en UTF-8).

  Chaque valeur est une chaîne de type : `string` (chaîne encodée en UTF-8).

  Contient des informations sur les paramètres actuels de votre cluster de bases de données. Par exemple, contient le paramètre actuel d'expiration des requêtes du cluster (`clusterQueryTimeoutInMs`).

- sparql : objet QueryLanguageVersion.

  Contient des informations sur le langage de requête SPARQL disponible sur votre cluster. Plus précisément, il contient un champ de version qui spécifie la version actuelle de SPARQL utilisée par le moteur.

- startTime : chaîne de type : `string` (chaîne encodée en UTF-8).

  Définie sur l'heure UTC (heure universelle coordonnée) à laquelle le processus serveur actuel a démarré.

- status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Définie sur `healthy` si l'instance ne rencontre aucun problème. Si l'instance est en cours de récupération suite à un incident ou parce qu'elle a été redémarrée et qu'il existe des transactions actives en cours d'exécution depuis le dernier arrêt de serveur, le statut est défini sur `recovery`.

Erreurs

- UnsupportedOperationException
- InternalFailureException
- ClientTimeoutException

- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- IllegalArgumentException
- TooManyRequestsException

ExecuteFastReset (action)

Le nom de la AWS CLI pour cette API est :`execute-fast-reset`.

L'API REST de réinitialisation rapide vous permet de réinitialiser un graphe Neptune rapidement et facilement, en supprimant toutes ses données.

La réinitialisation rapide de Neptune est un processus en deux étapes. Vous appelez d'abord `ExecuteFastReset` avec l'attribut `action` défini sur `initiateDatabaseReset`. Cette opération renvoie un jeton UUID que vous incluerez ensuite lors d'un nouvel appel d'`ExecuteFastReset` avec l'attribut `action` défini sur `performDatabaseReset`. Consultez Vider un cluster de bases de données Amazon Neptune à l'aide de l'API de réinitialisation rapide.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action ResetDatabaseneptune-db : IAM dans ce cluster.

Demande

- action (dans la CLI : `--action`) : obligatoire : action de type : `string` (chaîne encodée en UTF-8).

  Action de réinitialisation rapide. L'une des valeurs suivantes :

  - **`initiateDatabaseReset`** : cette action génère un jeton unique nécessaire pour effectuer réellement la réinitialisation rapide.
  - **`performDatabaseReset`** : cette action utilise le jeton généré par l'action `initiateDatabaseReset`pour effectuer réellement la réinitialisation rapide.

- token (dans la CLI : `--token`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Jeton de réinitialisation rapide permettant de lancer la réinitialisation.

Réponse

- payload : objet FastResetToken.

  La valeur `payload` n'est renvoyée que par l'action `initiateDatabaseReset` et contient le jeton unique à utiliser avec l'action `performDatabaseReset`pour effectuer la réinitialisation.
- status : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  La valeur `status` n'est renvoyée que pour l'action `performDatabaseReset` et indique si la demande de réinitialisation rapide est acceptée ou non.

Erreurs

- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- ServerShutdownException
- PreconditionsFailedException
- MethodNotAllowedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Structures de fonctionnement du moteur :

QueryLanguageVersion (structure)

Structure permettant d'exprimer la version du langage de requête.

Champs

- version : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

Version du langage de requête.

FastResetToken (structure)

Structure contenant le jeton utilisé pour initier une réinitialisation rapide.

Champs

- token : chaîne de type : `string` (chaîne encodée en UTF-8).

  UUID généré par la base de données lors de l'action `initiateDatabaseReset`, puis utilisé par `performDatabaseReset` pour réinitialiser la base de données.

Requête Neptune APIs

Actions de requête Gremlin :

- ExecuteGremlinQuery (action)
- ExecuteGremlinExplainQuery (action)
- ExecuteGremlinProfileQuery (action)
- ListGremlinQueries (action)
- GetGremlinQueryStatus (action)
- CancelGremlinQuery (action)

Actions de requête openCypher :

- ExecuteOpenCypherQuery (action)
- ExecuteOpenCypherExplainQuery (action)
- ListOpenCypherQueries (action)
- GetOpenCypherQueryStatus (action)
- CancelOpenCypherQuery (action)

Structures des requêtes :

- QueryEvalStats (structure)

- GremlinQueryStatus (structure)
- GremlinQueryStatusAttributes (structure)

ExecuteGremlinQuery (action)

Le nom de la AWS CLI pour cette API est :`execute-gremlin-query`.

Cette commande exécute une requête Gremlin. Amazon Neptune est compatible avec Apache TinkerPop 3 et Gremlin. Vous pouvez donc utiliser le langage de traversée Gremlin pour interroger le graphe, comme décrit dans la section The Graph de la documentation d'Apache 3. TinkerPop Plus d'informations sont également disponibles dans Accès à un graphe Neptune avec Gremlin.

Lorsque vous invoquez cette opération dans un cluster Neptune pour lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant les actions IAM suivantes dans ce cluster, en fonction de la requête :

- base de données Neptune : ReadDataViaQuery
- base de données Neptune : WriteDataViaQuery
- base de données Neptune : DeleteDataViaQuery

Notez que la clé de condition neptune-db ::Garmlin QueryLanguage IAM peut être utilisée dans le document de politique pour restreindre l'utilisation des requêtes Gremlin (voir Clés de condition disponibles dans les déclarations de politique d'accès aux données de Neptune IAM).

Demande

- gremlinQuery (dans la CLI : `--gremlin-query`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  À l'aide de cette API, vous pouvez exécuter des requêtes Gremlin au format chaîne comme vous pouvez le faire avec le point de terminaison HTTP. L'interface est compatible n'importe quelle version Gremlin utilisée par le cluster de bases de données (consultez la section du client Tinkerpop pour déterminer les versions de Gremlin prises en charge par votre version de moteur).
- serializer (dans la CLI : `--serializer`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si la valeur du paramètre est autre que null, les résultats de requête sont renvoyés dans un message de réponse sérialisé au format spécifié par ce paramètre. Consultez la section GraphSon de la TinkerPop documentation pour obtenir la liste des formats actuellement pris en charge.

Réponse

- meta : document de type : `document` (contenu ouvert indépendant du protocole représenté par un modèle de données de type JSON).

  Métadonnées relatives à la requête Gremlin.
- requestId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de la requête Gremlin.
- result : document de type : `document` (contenu ouvert indépendant du protocole représenté par un modèle de données de type JSON).

  Requête Gremlin générée par le serveur.
- status : objet GremlinQueryStatusAttributes.

  Statut de la requête Gremlin.

Erreurs

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException

- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteGremlinExplainQuery (action)

Le nom de la AWS CLI pour cette API est :`execute-gremlin-explain-query`.

Exécute une requête Gremlin Explain.

Amazon Neptune a ajouté une fonctionnalité Gremlin nommée `explain` qui fournit un outil en libre-service permettant de comprendre l'approche d'exécution adoptée par le moteur Neptune pour la requête. Vous l'appelez en ajoutant un paramètre `explain` à un appel HTTP qui soumet une requête Gremlin.

La fonctionnalité explain fournit des informations sur la structure logique des plans d'exécution de requête. Vous pouvez utiliser ces informations pour identifier les goulots d'étranglement potentiels liés à l'évaluation et à l'exécution et pour régler votre requête, comme expliqué dans la section Réglage des requêtes Gremlin. Vous pouvez aussi utiliser des indicateurs de requête pour améliorer les plans d'exécution de requêtes.

Lorsque vous invoquez cette opération dans un cluster Neptune pour lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique qui autorise les actions IAM suivantes dans ce cluster, en fonction de la requête :

- base de données Neptune : ReadDataViaQuery
- base de données Neptune : WriteDataViaQuery
- base de données Neptune : DeleteDataViaQuery

Notez que la clé de condition neptune-db ::Garmlin QueryLanguage IAM peut être utilisée dans le document de politique pour restreindre l'utilisation des requêtes Gremlin (voir Clés de condition disponibles dans les déclarations de politique d'accès aux données de Neptune IAM).

Demande

- gremlinQuery (dans la CLI : `--gremlin-query`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Chaîne de requête Gremlin explain.

Réponse

- output— a ReportAsText, de type : `blob` (un bloc de données binaires non interprétées).

  Blob de texte contenant le résultat de la fonctionnalité explain de Gremlin, comme décrit dans la section Réglage des requêtes Gremlin.

Erreurs

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException

- ConcurrentModificationException

ExecuteGremlinProfileQuery (action)

Le nom de la AWS CLI pour cette API est :`execute-gremlin-profile-query`.

Effectue une requête Gremlin Profile, qui effectue une traversée Gremlin spécifique, collecte différentes métriques sur l'exécution et génère en sortie un rapport de profil. Consultez API de profil Gremlin dans Neptune pour plus d'informations.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action ReadDataViaQueryneptune-db : IAM dans ce cluster.

Notez que la clé de condition neptune-db ::Garmlin QueryLanguage IAM peut être utilisée dans le document de politique pour restreindre l'utilisation des requêtes Gremlin (voir Clés de condition disponibles dans les déclarations de politique d'accès aux données de Neptune IAM).

Demande

- chop (dans la CLI : `--chop`) : entier de type : `integer` (entier signé de 32 bits).

  Si la valeur du paramètre est différente de zéro, la chaîne de résultats est tronquée à ce nombre de caractères. Si la valeur est zéro, la chaîne contient tous les résultats.
- gremlinQuery (dans la CLI : `--gremlin-query`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Chaîne de requête Gremlin à profiler.
- indexOps (dans la CLI : `--index-ops`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Si cet indicateur est défini sur `TRUE`, les résultats comprennent un rapport détaillé de toutes les opérations d'index qui ont eu lieu au cours de l'exécution et de la sérialisation des requêtes.
- results (dans la CLI : `--results`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Si cet indicateur est défini sur `TRUE`, les résultats de la requête sont collectés et affichés dans le rapport de profil. En cas de valeur `FALSE`, seul le nombre de résultats s'affiche.

- serializer (dans la CLI : `--serializer`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Si la valeur du paramètre est différente de null, les résultats collectés sont renvoyés dans un message de réponse sérialisé au format spécifié par ce paramètre. Consultez l'API de profil Gremlin dans Neptune pour plus d'informations.

Réponse

- output— a ReportAsText, de type : `blob` (un bloc de données binaires non interprétées).

  Blob de texte contenant le résultat de Gremlin Profile. Consultez API de profil Gremlin dans Neptune pour plus d'informations.

Erreurs

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException

- MissingParameterException
- ConcurrentModificationException

ListGremlinQueries (action)

Le nom de la AWS CLI pour cette API est :`list-gremlin-queries`.

Répertorie les requêtes Gremlin actives. Consultez la section API de statut des requêtes Gremlin pour plus d'informations sur le résultat.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action GetQueryStatusneptune-db : IAM dans ce cluster.

Notez que la clé de condition neptune-db ::Garmlin QueryLanguage IAM peut être utilisée dans le document de politique pour restreindre l'utilisation des requêtes Gremlin (voir Clés de condition disponibles dans les déclarations de politique d'accès aux données de Neptune IAM).

Demande

- includeWaiting (dans la CLI : `--include-waiting`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Si ce paramètre est défini sur `TRUE`, la liste renvoyée inclut les requêtes en attente. La valeur par défaut est `FALSE`.

Réponse

- acceptedQueryCount : entier de type : `integer` (entier signé de 32 bits).

  Nombre de requêtes qui ont été acceptées mais qui n'ont pas encore été terminées, y compris les requêtes dans la file d'attente.
- queries : tableau d'objets GremlinQueryStatus.

  Liste des requêtes actuelles.
- runningQueryCount : entier de type : `integer` (entier signé de 32 bits).

  Nombre de requêtes Gremlin actuellement en cours d'exécution.

Erreurs

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

GetGremlinQueryStatus (action)

> Le nom de la AWS CLI pour cette API est :`get-gremlin-query-status`.

Génère le statut d'une requête Gremlin spécifiée.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action GetQueryStatusneptune-db : IAM dans ce cluster.

Notez que la clé de condition neptune-db ::Garmlin QueryLanguage IAM peut être utilisée dans le document de politique pour restreindre l'utilisation des requêtes Gremlin (voir Clés de condition disponibles dans les déclarations de politique d'accès aux données de Neptune IAM).

Demande

- queryId (dans la CLI : `--query-id`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de la requête Gremlin.

Réponse

- queryEvalStats : objet QueryEvalStats.

  Statut d'évaluation de la requête Gremlin.
- queryId : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la requête dont le statut est renvoyé.
- queryString : chaîne de type : `string` (chaîne encodée en UTF-8).

  Chaîne de requête Gremlin.

Erreurs

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException

- ConcurrentModificationException

CancelGremlinQuery (action)

Le nom de la AWS CLI pour cette API est :`cancel-gremlin-query`.

Annule une requête Gremlin. Pour plus d'informations, consultez la section Annulation de requêtes Gremlin.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action CancelQueryneptune-db : IAM dans ce cluster.

Demande

- queryId (dans la CLI : `--query-id`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de la requête à annuler.

Réponse

- status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Statut de l'annulation.

Erreurs

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException

- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

Actions de requête openCypher :

ExecuteOpenCypherQuery (action)

Le nom de la AWS CLI pour cette API est :`execute-open-cypher-query`.

Exécute une requête openCypher. Consultez Accès au graphe de Neptune avec openCypher pour plus d'informations.

Neptune prend en charge la création d'applications de graphe à l'aide d'openCypher, qui est actuellement l'un des langages de requête les plus populaires parmi les développeurs travaillant avec des bases de données orientées graphe. Les développeurs, les analystes et les scientifiques des données apprécient la syntaxe déclarative openCypher inspirée de SQL, car sa structure familière facilite l'interrogation des graphes de propriétés.

Le langage openCypher a été initialement développé par Neo4j, puis est devenu open source en 2015 et a contribué au projet openCypher sous une licence open source Apache 2.

Notez que lorsque vous invoquez cette opération dans un cluster Neptune pour lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique qui autorise les actions IAM suivantes dans ce cluster, en fonction de la requête :

- base de données Neptune : ReadDataViaQuery
- base de données Neptune : WriteDataViaQuery
- base de données Neptune : DeleteDataViaQuery

Notez également que la clé de condition neptune-db : QueryLanguage : OpenCypher IAM peut être utilisée dans le document de politique pour restreindre l'utilisation des requêtes OpenCypher (voir

Clés de condition disponibles dans les déclarations de politique d'accès aux données de Neptune IAM).

Demande

- openCypherQuery (dans la CLI : `--open-cypher-query`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Chaîne de requête openCypher à exécuter.
- parameters (dans la CLI : `--parameters`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Paramètres de requête openCypher pour l'exécution des requêtes. Pour plus d'informations, consultez Exemples de requêtes paramétrées openCypher.

Réponse

- results : obligatoire : document de type : `document` (contenu ouvert indépendant du protocole représenté par un modèle de données de type JSON).

  Résultats de la requête openCypher.

Erreurs

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException

- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteOpenCypherExplainQuery (action)

Le nom de la AWS CLI pour cette API est :`execute-open-cypher-explain-query`.

Exécute une demande `explain` openCypher. Pour plus d'informations, consultez Fonctionnalité explain d'openCypher.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action ReadDataViaQueryneptune-db : IAM dans ce cluster.

Notez que la clé de condition neptune-db : QueryLanguage : OpenCypher IAM peut être utilisée dans le document de politique pour restreindre l'utilisation des requêtes OpenCypher (voir Clés de condition disponibles dans les déclarations de politique d'accès aux données de Neptune IAM).

Demande

- explainMode(dans la CLI :`--explain-mode`) — Obligatoire : an OpenCypherExplainMode, de type : `string` (une chaîne codée en UTF-8).

  Mode `explain` openCypher. Peut avoir l'une des valeurs suivantes : `static`, `dynamic` ou `details`.
- openCypherQuery (dans la CLI : `--open-cypher-query`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Chaîne de requête openCypher.
- parameters (dans la CLI : `--parameters`) : chaîne de type : `string` (chaîne encodée en UTF-8).

Paramètres de requête openCypher.

Réponse

- results : obligatoire : blob de type : `blob` (bloc de données binaires non interprétées).

  Blob de texte contenant les résultats `explain` openCypher.

Erreurs

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ListOpenCypherQueries (action)

Le nom de la AWS CLI pour cette API est :`list-open-cypher-queries`.

Répertorie les requêtes openCypher actives. Consultez la section Point de terminaison d'état Neptune openCypher pour plus d'informations.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action GetQueryStatusneptune-db : IAM dans ce cluster.

Notez que la clé de condition neptune-db : QueryLanguage : OpenCypher IAM peut être utilisée dans le document de politique pour restreindre l'utilisation des requêtes OpenCypher (voir Clés de condition disponibles dans les déclarations de politique d'accès aux données de Neptune IAM).

Demande

- includeWaiting (dans la CLI : `--include-waiting`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Lorsque ce paramètre est défini sur `TRUE` et qu'aucun autre paramètre n'est présent, des informations d'état sont renvoyées pour les requêtes en attente ainsi que pour les requêtes en cours d'exécution.

Réponse

- acceptedQueryCount : entier de type : `integer` (entier signé de 32 bits).

  Nombre de requêtes qui ont été acceptées mais qui n'ont pas encore été terminées, y compris les requêtes dans la file d'attente.
- queries : tableau d'objets GremlinQueryStatus.

  Liste des requêtes openCypher en cours.
- runningQueryCount : entier de type : `integer` (entier signé de 32 bits).

  Nombre de requêtes openCypher en cours d'exécution.

Erreurs

- InvalidNumericDataException

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

GetOpenCypherQueryStatus (action)

Le nom de la AWS CLI pour cette API est :`get-open-cypher-query-status`.

Récupère le statut d'une requête openCypher spécifiée.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action GetQueryStatusneptune-db : IAM dans ce cluster.

Notez que la clé de condition neptune-db : QueryLanguage : OpenCypher IAM peut être utilisée dans le document de politique pour restreindre l'utilisation des requêtes OpenCypher (voir Clés de condition disponibles dans les déclarations de politique d'accès aux données de Neptune IAM).

Demande

- queryId (dans la CLI : `--query-id`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

ID unique de la requête openCypher pour laquelle le statut de la requête doit être récupéré.

Réponse

- queryEvalStats : objet QueryEvalStats.

  Statut de l'évaluation de la requête openCypher.
- queryId : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID unique de la requête dont le statut est renvoyé.
- queryString : chaîne de type : `string` (chaîne encodée en UTF-8).

  Chaîne de requête openCypher.

Erreurs

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

CancelOpenCypherQuery (action)

Le nom de la AWS CLI pour cette API est :`cancel-open-cypher-query`.

Annule une requête openCypher spécifiée. Consultez la section Point de terminaison d'état Neptune openCypher pour plus d'informations.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action CancelQueryneptune-db : IAM dans ce cluster.

Demande

- queryId (dans la CLI : `--query-id`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID unique de la requête openCypher à annuler.
- silent (dans la CLI : `--silent`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Si ce paramètre est défini sur `TRUE`, l'annulation de la requête openCypher se produit silencieusement.

Réponse

- payload : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Charge utile d'annulation pour la requête openCypher.
- status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Statut d'annulation de la requête openCypher.

Erreurs

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException

- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

Structures des requêtes :

QueryEvalStats (structure)

Structure permettant de saisir les statistiques relatives aux requêtes, telles que le nombre de requêtes en cours, acceptées ou en attente, ainsi que leurs détails.

Champs

- cancelled : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Valeur définie sur TRUE si la requête a été annulée, ou sur FALSE dans le cas contraire.
- elapsed : entier de type : `integer` (entier signé de 32 bits).

  Nombre de microsecondes d'exécution de la requête jusqu'ici.
- subqueries : document de type : `document` (contenu ouvert indépendant du protocole représenté par un modèle de données de type JSON).

  Nombre de sous-requêtes de cette requête.
- waited : entier de type : `integer` (entier signé de 32 bits).

  Indique le temps d'attente de la requête, en millisecondes.

GremlinQueryStatus (structure)

Capture le statut d'une requête Gremlin (voir la page API de statut des requêtes Gremlin).

Champs

- queryEvalStats : objet QueryEvalStats.

  Statistiques de la requête Gkremlin.
- queryId : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la requête Gremlin.
- queryString : chaîne de type : `string` (chaîne encodée en UTF-8).

  Chaîne de la requête Gremlin.

GremlinQueryStatusAttributes (structure)

Contient les composants de statut d'une requête Gremlin.

Champs

- attributes : document de type : `document` (contenu ouvert indépendant du protocole représenté par un modèle de données de type JSON).

  Attributs du statut de la requête Gremlin.
- code : entier de type : `integer` (entier signé de 32 bits).

  Code de réponse HTTP renvoyé par la demande de requête Gremlin.
- message : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message d'état.

API de chargeur en bloc du plan de données Neptune

Bulk-load actions :

- StartLoaderJob (action)

- GetLoaderJobStatus (action)
- ListLoaderJobs (action)
- CancelLoaderJob (action)

Structure de chargement en bloc :

- LoaderIdResult (structure)

StartLoaderJob (action)

Le nom de la AWS CLI pour cette API est :`start-loader-job`.

Démarre une tâche de chargeur en bloc Neptune pour charger les données d'un compartiment Amazon S3 dans une instance de base de données Neptune. Consultez la section Utilisation du chargeur en bloc Amazon Neptune pour ingérer des données.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action StartLoaderJobneptune-db : IAM dans ce cluster.

Demande

- dependencies(dans la CLI :`--dependencies`) — une chaîne de type : `string` (chaîne UTF-8 codée).

  Il s'agit d'un paramètre facultatif qui peut subordonner une demande de chargement en file d'attente à la réussite d'une ou de plusieurs tâches précédentes dans la file d'attente.

  Neptune peut mettre en file d'attente jusqu'à 64 requêtes de chargement à la fois, si leurs paramètres `queueRequest` sont définis sur `"TRUE"`. Le paramètre `dependencies` vous permet de rendre l'exécution d'une telle requête en file d'attente dépendante de la réussite d'une ou de plusieurs requêtes précédentes spécifiées dans la file d'attente.

  Par exemple, si les charges `Job-A` et `Job-B` sont indépendantes l'une de l'autre, mais que la charge `Job-C` a besoin que `Job-A` et `Job-B` soient terminées avant de commencer, procédez comme suit :

  1. Soumettez `load-job-A` et `load-job-B` l'une après l'autre dans n'importe quel ordre, et enregistrez leurs ID de chargement.

2. Soumettez `load-job-C` avec les ID de chargement des deux tâches dans son domaine `dependencies` :

   Example

   ```
   "dependencies" : ["(job_A_load_id)", "(job_B_load_id)"]
   ```

   En raison du paramètre `dependencies`, le chargeur en bloc ne démarrera pas `Job-C` avant que `Job-A` et `Job-B` soient terminées avec succès. Si l'un d'entre eux échoue, il ne Job-C sera pas exécuté et son statut sera défini sur`LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

   Vous pouvez configurer plusieurs niveaux de dépendance de cette façon, de sorte que l'échec d'une tâche entraîne l'annulation de toutes les demandes qui en dépendent directement ou indirectement.

- failOnError (dans la CLI : `--fail-on-error`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  `failOnError` : indicateur permettant d'activer un arrêt complet au niveau d'une erreur.

  Valeurs autorisées : `"TRUE"`, `"FALSE"`.

  Valeur par défaut : `"TRUE"`.

  Lorsque ce paramètre est défini sur `"FALSE"`, le chargeur essaie de charger toutes les données à l'emplacement spécifié, en ignorant les entrées contenant des erreurs.

  Lorsque ce paramètre est défini sur `"TRUE"`, le chargeur s'arrête dès qu'il rencontre une erreur. Les données chargées jusqu'à ce point persistent.

- format(dans la CLI :`--format`) — Obligatoire : un format, de type : `string` (une chaîne UTF-8 codée).

  Format des données. Pour plus d'informations sur les formats de données pour la commande `Loader` Neptune, consultez Formats de chargement de données.

  Valeurs autorisées

  - **`csv`** pour le format de données CSV Gremlin.
  - **`opencypher`** pour le format de données CSV openCypher.
  - **`ntriples`**pour le format de données N-Triples RDF.

  - **`nquads`**pour le format de données N-Quads RDF.
  - **`rdfxml`** pour le format de données RDF RDF/XML.
  - **`turtle`** pour le format de données RDF Turtle.
- iamRoleArn(dans la CLI :`--iam-role-arn`) — Obligatoire : une chaîne, de type : `string` (une chaîne UTF-8 codée).

  Amazon Resource Name (ARN) d'un rôle IAM qui doit être endossé par l'instance de base de données Neptune pour accéder au compartiment S3. L'ARN du rôle IAM fourni ici doit être attaché au cluster de bases de données (consultez Ajout du rôle IAM à un cluster Amazon Neptune).
- mode(dans la CLI :`--mode`) — un mode, de type : `string` (une chaîne UTF-8 codée).

  Mode de tâche de chargement.

  Valeurs autorisées : `RESUME`, `NEW`, `AUTO`

  Valeur par défaut : `AUTO`.

  - `RESUME` : en mode REPRISE, le chargeur recherche un chargement précédent à partir de cette source et, s'il en trouve un, reprend cette tâche de chargement. Si aucune tâche de chargement précédente n'est trouvée, le chargeur s'arrête.

    Le chargeur évite de recharger les fichiers qui ont été chargés avec succès lors d'une tâche précédente. Il essaie uniquement de traiter les fichiers ayant échoué. Si vous aviez supprimé des données précédemment chargées à partir de votre cluster Neptune, ces données ne sont pas rechargées dans ce mode. Si une tâche de chargement précédente a chargé tous les fichiers de la même source avec succès, rien n'est rechargé, et le chargeur renvoie une réussite.
  - `NEW` : le mode NOUVEAU crée une autre demande de chargement sans tenir compte des chargements précédents. Vous pouvez utiliser ce mode pour recharger toutes les données d'une source après la suppression de données précédemment chargées à partir de votre cluster Neptune ou pour charger de nouvelles données disponibles sur la même source.
  - `AUTO` : en mode AUTO, le chargeur recherche une tâche de chargement précédente à partir de la même source, et s'il en trouve une, reprend cette tâche, exactement comme en mode `RESUME`.

    Si le chargeur ne trouve pas de tâche de chargement précédente à partir de la même source, il charge toutes les données de la source, exactement comme en mode `NEW`.

- parallelism(dans la CLI :`--parallelism`) — un parallélisme, de type : `string` (une chaîne UTF-8 codée).

  Le paramètre facultatif `parallelism` peut être défini de façon à réduire le nombre de threads utilisés par le processus de chargement en bloc.

  Valeurs autorisées :

  - `LOW` : le nombre de threads utilisés est le nombre de vCPU disponibles divisé par huit.
  - `MEDIUM` : le nombre de threads utilisés est le nombre de vCPU disponibles divisé par deux.
  - `HIGH` : le nombre de threads utilisés est le même que le nombre de vCPU disponibles.
  - `OVERSUBSCRIBE` : le nombre de threads utilisés est le nombre de vCPU disponibles multiplié par deux. Si cette valeur est utilisée, le chargeur en bloc accepte toutes les ressources disponibles.

    Cela ne signifie toutefois pas que le paramètre `OVERSUBSCRIBE` entraîne une utilisation du CPU à 100 %. L'opération de chargement étant limitée, I/O l'utilisation maximale du processeur à laquelle on peut s'attendre se situe entre 60 % et 70 %.

  Valeur par défaut : `HIGH`

  Le paramètre `parallelism` peut parfois entraîner un blocage entre les threads lors du chargement des données openCypher. Lorsque cela se produit, Neptune renvoie le message d'erreur `LOAD_DATA_DEADLOCK`. Vous pouvez généralement résoudre le problème en définissant un paramètre `parallelism` inférieur et en soumettant la commande de chargement à une nouvelle tentative.
- parserConfiguration (dans la CLI : `--parser-configuration`) : tableau de mappage de paires clé-valeur où :

  Chaque clé est une chaîne de type : `string` (chaîne UTF-8 codée).

  Chaque valeur est une chaîne de type : `string` (chaîne UTF-8 codée).

  `parserConfiguration` : objet facultatif avec des valeurs de configuration d'analyseur supplémentaires. Chacun des paramètres enfants est également facultatif :

  - **`namedGraphUri`** : graphe par défaut pour tous les formats RDF quand aucun graphe n'est spécifié (pour les formats autres que quadruplets et les entrées NQUAD sans graphe).

La valeur par défaut est `https://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

- **`baseUri`**— L'URI de base pour les formats RDF/XML et Turtle.

  La valeur par défaut est `https://aws.amazon.com/neptune/default`.

- **`allowEmptyStrings`** : les utilisateurs de Gremlin doivent être en mesure de transmettre des valeurs de chaîne vides ("") en tant que propriétés de nœud et d'arête lors du chargement de données CSV. Si la valeur `allowEmptyStrings` est définie sur `false` (valeur par défaut), ces chaînes vides sont traitées comme des valeurs nulles et ne sont pas chargées.

  Si la valeur `allowEmptyStrings` est définie sur `true`, le chargeur traite les chaînes vides comme des valeurs de propriété valides et les charge en conséquence.

- queueRequest (dans la CLI : `--queue-request`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Il s'agit d'un paramètre d'indicateur facultatif qui indique si la demande de chargement peut être mise en file d'attente ou non.

  Vous n'avez pas besoin d'attendre qu'une tâche de chargement soit terminée avant d'émettre la suivante, car Neptune peut mettre en file d'attente jusqu'à 64 tâches à la fois, si leurs paramètres `queueRequest` sont tous définis sur `"TRUE"`. L'ordre des tâches en file d'attente repose sur une structure FIFO (premier entré, premier sorti).

  Si le paramètre `queueRequest` est omis ou défini sur `"FALSE"`, la demande de chargement échouera si une autre tâche de chargement est déjà en cours d'exécution.

  Valeurs autorisées : `"TRUE"`, `"FALSE"`.

  Valeur par défaut : `"FALSE"`.

- s3BucketRegion(dans la CLI :`--s-3-bucket-region`) — Obligatoire : a S3BucketRegion, de type : `string` (une chaîne UTF-8 codée).

  Région Amazon du compartiment S3. Elle doit correspondre à la région Amazon du cluster de bases de données.

- source(dans la CLI :`--source`) — Obligatoire : une chaîne, de type : `string` (une chaîne UTF-8 codée).

Le paramètre `source` accepte un URI S3 qui identifie un seul fichier, plusieurs fichiers, un dossier ou plusieurs dossiers. Neptune charge tous les fichiers de données dans n'importe quel dossier spécifié.

L'URI peut être à l'un des formats suivants.

- `s3://(bucket_name)/(object-key-name)`
- `https://s3.amazonaws.com/(bucket_name)/(object-key-name)`
- `https://s3.us-east-1.amazonaws.com/(bucket_name)/(object-key-name)`

L'`object-key-name`élément de l'URI est équivalent au paramètre de préfixe dans un appel d'ListObjectsAPI S3. Il identifie tous les objets du compartiment S3 spécifié dont le nom commence par ce préfixe. Il peut s'agir d'un seul fichier ou dossier, ou de plusieurs and/or dossiers de fichiers.

Le dossier spécifié peut contenir plusieurs fichiers de sommet et plusieurs fichiers d'arête.

- updateSingleCardinalityProperties (dans la CLI : `--update-single-cardinality-properties`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  `updateSingleCardinalityProperties` est un paramètre facultatif qui contrôle la façon dont le chargeur en bloc traite une nouvelle valeur pour les propriétés de sommet ou d'arête à cardinalité unique.

  Valeurs autorisées : `"TRUE"`, `"FALSE"`.

  Valeur par défaut : `"FALSE"`.

  Par défaut, ou lorsque `updateSingleCardinalityProperties` est explicitement défini sur `"FALSE"`, le chargeur traite une nouvelle valeur comme une erreur, car elle ne respecte pas la cardinalité unique.

  En revanche, lorsque `updateSingleCardinalityProperties` est défini sur `"TRUE"`, le chargeur en bloc remplace la valeur existante par la nouvelle. Si plusieurs valeurs de propriété de sommet à cardinalité unique ou d'arête sont fournies dans le ou les fichiers source en cours de chargement, la valeur finale à l'issue du chargement en bloc peut être une de ces nouvelles valeurs. Le chargeur garantit uniquement que la valeur existante a été remplacée par une des nouvelles.
- userProvidedEdgeIds (dans la CLI : `--user-provided-edge-ids`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

Ce paramètre n'est obligatoire que lors du chargement de données openCypher contenant des ID de relation. Il doit être inclus et défini sur `True` lorsque les ID de relation openCypher sont explicitement fournis dans les données de chargement (recommandé).

Quand `userProvidedEdgeIds` est absent ou défini sur `True`, une colonne `:ID` doit être présente dans chaque fichier de relations inclus dans le chargement.

Quand `userProvidedEdgeIds` est présent et défini sur `False`, les fichiers de relations inclus dans le chargement ne doivent pas contenir de colonne `:ID`. Au lieu de cela, le chargeur Neptune génère automatiquement un ID pour chaque relation.

Il est utile de fournir des ID de relation de manière explicite afin que le chargeur puisse reprendre le chargement après correction d'une erreur dans les données CSV, sans avoir à recharger les relations déjà chargées. Si aucun ID de relation n'a été attribué explicitement, le chargeur ne peut pas reprendre un chargement défaillant si un fichier de relation a dû être corrigé, et doit à la place recharger toutes les relations.

Réponse

- payload : obligatoire : tableau de mappage de paires clé-valeur où :

  Chaque clé est une chaîne de type : `string` (chaîne UTF-8 codée).

  Chaque valeur est une chaîne de type : `string` (chaîne UTF-8 codée).

  Contient une paire nom-valeur `loadId` qui fournit un identifiant pour l'opération de chargement.
- status— Obligatoire : une chaîne de type : `string` (chaîne UTF-8 codée).

  Code de retour HTTP indiquant le statut de la tâche de chargement.

Erreurs

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException

- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- S3Exception
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

GetLoaderJobStatus (action)

Le nom de la AWS CLI pour cette API est :`get-loader-job-status`.

Génère des informations de statut sur une tâche de chargement spécifiée. Neptune conserve une trace des 1 024 tâches de chargement en bloc les plus récentes et stocke les 10 000 dernières informations détaillées d'erreur par tâche.

Consultez l' Get-Status API Neptune Loader pour plus d'informations.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action GetLoaderJobStatusneptune-db : IAM dans ce cluster.

Demande

- details (dans la CLI : `--details`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indicateur signalant s'il faut ou non inclure des détails autres que le statut général (`TRUE` ou `FALSE` ; la valeur par défaut est `FALSE`).
- errors (dans la CLI : `--errors`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indicateur signalant s'il faut ou non inclure une liste des erreurs rencontrées (`TRUE` ou `FALSE` ; la valeur par défaut est `FALSE`).

La liste d'erreurs est paginée. Les paramètres `page` et `errorsPerPage` vous permettent de parcourir toutes les erreurs.

- errorsPerPage(dans la CLI :`--errors-per-page`) — a PositiveInteger, de type : `integer` (un entier signé de 32 bits), au moins 1 ? st ?.

  Nombre d'erreurs renvoyées sur chaque page (entier positif ; la valeur par défaut est `10`). Valide uniquement si le paramètre `errors` est défini sur `TRUE`.

- loadId(dans la CLI :`--load-id`) — Obligatoire : une chaîne, de type : `string` (une chaîne UTF-8 codée).

  ID de chargement de la tâche de chargement dont vous souhaitez obtenir le statut.

- page(dans la CLI :`--page`) — a PositiveInteger, de type : `integer` (un entier signé de 32 bits), au moins 1 ? st ?.

  Numéro de la page d'erreur (entier positif ; la valeur par défaut est 1). Valide uniquement lorsque le paramètre `errors` est défini sur `TRUE`.

Réponse

- payload— Obligatoire : un document, du type : `document` (un contenu ouvert indépendant du protocole représenté par un JSON-like modèle de données).

  Informations sur le statut de la tâche de chargement, dans une mise en page qui pourrait ressembler à ceci :

  Example

```
{
          "status" : "200 OK",
          "payload" : {
            "feedCount" : [
              {
                "LOAD_FAILED" : (number)
              }
            ],
            "overallStatus" : {
              "fullUri" : "s3://(bucket)/(key)",
              "runNumber" : (number),
              "retryNumber" : (number),
```

```
            "status" : "(string)",
            "totalTimeSpent" : (number),
            "startTime" : (number),
            "totalRecords" : (number),
            "totalDuplicates" : (number),
            "parsingErrors" : (number),
            "datatypeMismatchErrors" : (number),
            "insertErrors" : (number),
          },
          "failedFeeds" : [
            {
              "fullUri" : "s3://(bucket)/(key)",
              "runNumber" : (number),
              "retryNumber" : (number),
              "status" : "(string)",
              "totalTimeSpent" : (number),
              "startTime" : (number),
              "totalRecords" : (number),
              "totalDuplicates" : (number),
              "parsingErrors" : (number),
              "datatypeMismatchErrors" : (number),
              "insertErrors" : (number),
            }
          ],
          "errors" : {
            "startIndex" : (number),
            "endIndex" : (number),
            "loadId" : "(string),
            "errorLogs" : [ ]
          }
        }
      }
```

- status— Obligatoire : une chaîne de type : `string` (chaîne UTF-8 codée).

  Code de réponse HTTP de la demande.

Erreurs

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException

- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ListLoaderJobs (action)

Le nom de la AWS CLI pour cette API est :`list-loader-jobs`.

Récupère la liste des `loadIds` pour toutes les tâches de chargement actives.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action ListLoaderJobsneptune-db : IAM dans ce cluster.

Demande

- includeQueuedLoads (dans la CLI : `--include-queued-loads`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Paramètre facultatif qui peut être utilisé pour exclure les ID de chargement des demandes de chargement en file d'attente lorsqu'une liste d'ID de chargement est demandée en définissant ce paramètre sur `FALSE`. La valeur par défaut est `TRUE`.
- limit(dans la CLI :`--limit`) — a ListLoaderJobsInputLimitInteger, de type : `integer` (un entier signé de 32 bits), pas moins de 1 ou plus de 100 ? 1er ? s.

  Nombre d'ID de chargement à répertorier. Doit être un entier positif supérieur à zéro et inférieur ou égal à `100` (valeur par défaut).

Réponse

- payload : obligatoire : objet LoaderIdResult.

  Liste des ID de tâche demandés.

- status— Obligatoire : une chaîne de type : `string` (chaîne UTF-8 codée).

  Renvoie le statut de la demande d'affichage des tâches.

Erreurs

- UnsupportedOperationException
- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- InternalFailureException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException

CancelLoaderJob (action)

Le nom de la AWS CLI pour cette API est :`cancel-loader-job`.

Annule une tâche de chargement spécifiée. Il s'agit d'une demande HTTP `DELETE`. Consultez l' Get-Status API Neptune Loader pour plus d'informations.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action CancelLoaderJobneptune-db : IAM dans ce cluster.

Demande

- loadId(dans la CLI :`--load-id`) — Obligatoire : une chaîne, de type : `string` (une chaîne UTF-8 codée).

  ID de la tâche à supprimer.

Réponse

- status— une chaîne, de type : `string` (chaîne UTF-8 codée).

  Statut de l'annulation.

Erreurs

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Structure de chargement en bloc :

LoaderIdResult (structure)

Contient une liste d'ID de chargement.

Champs

- loadIds— Il s'agit d'une chaîne de type : `string` (chaîne UTF-8 codée).

  Liste des ID de chargement.

API de plan de données des flux Neptune

Actions d'accès aux flux :

- GetPropertygraphStream (action)

Structures de données de flux :

- PropertygraphRecord (structure)
- PropertygraphData (structure)

GetPropertygraphStream (action)

Le nom de la AWS CLI pour cette API est :`get-propertygraph-stream`.

Obtient un flux pour un graphe de propriétés.

La fonctionnalité Neptune Streams vous permet de générer une séquence complète d'entrées de journal des modifications qui enregistrent chaque modification apportée à vos données de graphe au fur et à mesure de leur application. `GetPropertygraphStream` vous permet de collecter ces entrées de journaux de modifications pour un graphe de propriétés.

La fonctionnalité Neptune Streams doit être activée sur votre Neptune. DBcluster Pour activer les flux, définissez le paramètre de cluster de bases de données neptune_streams sur 1.

Consultez Capture des modifications de graphe en temps réel à l'aide des flux Neptune.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action GetStreamRecordsneptune-db : IAM dans ce cluster.

Lorsque vous invoquez cette opération dans un cluster Neptune pour lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'une des actions IAM suivantes, en fonction de la requête :

Notez que vous pouvez restreindre les requêtes de graphes de propriétés à l'aide des clés de contexte IAM suivantes :

- neptune-db ::Gremlin QueryLanguage
- base de données Neptune : QueryLanguage OpenCypher

Consultez Clés de condition disponibles dans les déclarations de stratégie d'accès aux données IAM Neptune.

Demande

- commitNum (dans la CLI : `--commit-num`) : long de type : `long` (entier signé de 64 bits).

  Numéro de validation de l'enregistrement de départ à lire à partir du flux du journal des modifications. Ce paramètre est obligatoire quand `iteratorType` est `AT_SEQUENCE_NUMBER` ou `AFTER_SEQUENCE_NUMBER`, et ignoré quand `iteratorType` indique `TRIM_HORIZON` ou `LATEST`.
- encoding (dans la CLI : `--encoding`) : encodage de type : `string` (chaîne encodée en UTF-8).

  Si ce paramètre est défini sur TRUE, Neptune compresse la réponse à l'aide de l'encodage gzip.
- iteratorType(dans la CLI :`--iterator-type`) — un IteratorType, de type : `string` (une chaîne codée en UTF-8).

  Les valeurs suivantes sont possibles :

  - `AT_SEQUENCE_NUMBER` : indique que la lecture doit commencer à partir du numéro de séquence d'événement spécifié conjointement par les paramètres `commitNum` et `opNum`.
  - `AFTER_SEQUENCE_NUMBER` : indique que la lecture doit commencer juste après le numéro de séquence d'événement spécifié conjointement par les paramètres `commitNum` et `opNum`.
  - `TRIM_HORIZON` : indique que la lecture doit commencer au niveau du dernier enregistrement non tronqué du système, qui est le plus ancien enregistrement n'ayant pas expiré (pas encore supprimé) dans le flux de journaux des modifications.

  - `LATEST` : indique que la lecture doit commencer au niveau de l'enregistrement le plus récent dans le système, qui est le dernier enregistrement n'ayant pas expiré (pas encore supprimé) dans le flux de journaux des modifications.
- limit(dans la CLI :`--limit`) — a GetPropertygraphStreamInputLimitLong, de type : `long` (un entier signé de 64 bits), pas moins de 1 ou plus de 100 000 ? 1er ? s.

  Spécifie le nombre maximal d'enregistrements à renvoyer. Il existe également une limite de taille de 10 Mo pour la réponse qui ne peut pas être modifiée et qui est prioritaire sur le nombre d'enregistrements spécifié dans le paramètre `limit`. La réponse inclut un enregistrement de dépassement de seuil si la limite de 10 Mo a été atteinte.

  La plage de valeurs `limit` est comprise entre 1 et 100 000, avec une valeur par défaut de 10.
- opNum (dans la CLI : `--op-num`) : long de type : `long` (entier signé de 64 bits).

  Numéro de séquence d'opération au sein de la validation spécifiée à partir duquel commencer la lecture dans les données du flux du journal des modifications. L'argument par défaut est 1.

Réponse

- format : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Format de sérialisation pour les enregistrements de modification renvoyés. À l'heure actuelle, la seule valeur prise en charge est `PG_JSON`.
- lastEventId : obligatoire : tableau de mappage de paires clé-valeur où :

    Chaque clé est une chaîne de type : `string` (chaîne encodée en UTF-8).

    Chaque valeur est une chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de séquence de la dernière modification dans la réponse du flux.

  Un ID d'événement se compose de deux champs : `commitNum` qui identifie une transaction ayant modifié le graphe et `opNum` qui identifie une opération spécifique au sein de cette transaction.

  Example

```
"eventId": {
          "commitNum": 12,
          "opNum": 1
```

```
        }
```

- lastTrxTimestampInMillis : obligatoire : long facultatif de type : `long` (entier signé de 64 bits).

  Heure à laquelle la validation de la transaction a été demandée, en millisecondes, à partir de l'Unix Epoch.
- records : obligatoire : tableau d'objets PropertygraphRecord.

  Tableau des enregistrements du flux de journaux des modifications sérialisés inclus dans la réponse.
- totalRecords : obligatoire : entier de type : `integer` (entier signé de 32 bits).

  Nombre total d'enregistrements dans la réponse.

Erreurs

- UnsupportedOperationException
- ExpiredStreamException
- InvalidParameterException
- MemoryLimitExceededException
- StreamRecordsNotFoundException
- ClientTimeoutException
- PreconditionsFailedException
- ThrottlingException
- ConstraintViolationException
- InvalidArgumentException
- IllegalArgumentException
- TooManyRequestsException

Structures de données de flux :

PropertygraphRecord (structure)

Structure d'un enregistrement de graphe de propriétés.

Champs

- commitTimestampInMillis : obligatoire : long facultatif de type : `long` (entier signé de 64 bits).

  Heure à laquelle la validation de la transaction a été demandée, en millisecondes, à partir de l'Unix Epoch.
- data : obligatoire : objet PropertygraphData.

  Enregistrement de modification sérialisé Gremlin, SPARQL ou openCypher.
- eventId : obligatoire : tableau de mappage de paires clé-valeur où :

  Chaque clé est une chaîne de type : `string` (chaîne encodée en UTF-8).

  Chaque valeur est une chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant de séquence de l'enregistrement de modification du flux.
- isLastOp : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Présent uniquement si cette opération est la dernière dans sa transaction. S'il est présent, il est défini sur true. Utile pour s'assurer qu'une transaction est consommée dans son intégralité.
- op : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Opération à l'origine de la modification.

PropertygraphData (structure)

Enregistrement de modification sérialisé Gremlin ou openCypher.

Champs

- from : chaîne de type : `string` (chaîne encodée en UTF-8).

  S'il s'agit d'une arête (type = `e`), ID du sommet `from` ou du nœud source correspondant.
- id : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de l'élément Gremlin ou openCypher.
- key : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la propriété. Pour les étiquettes d'élément, il s'agit de `label`.

- to : chaîne de type : `string` (chaîne encodée en UTF-8).

  S'il s'agit d'une arête (type = `e`), ID du sommet `to` ou du nœud cible correspondant.
- type : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de cet élément Gremlin ou openCypher. Doit être l'une des valeurs suivantes :
  - **v1** : étiquette du sommet pour Gremlin ou étiquette du nœud pour openCypher.
  - **vp** : propriétés du sommet pour Gremlin ou propriétés du nœud pour openCypher.
  - **e** : arête et étiquette d'arête pour Gremlin ou relation et type de relation pour openCypher.
  - **ep** : propriétés de l'arête pour Gremlin ou propriétés de la relation pour openCypher.
- value : obligatoire : document de type : `document` (contenu ouvert indépendant du protocole représenté par un modèle de données de type JSON).

  Il s'agit d'un objet JSON qui contient un champ pour la valeur elle-même et un champ pour le type de données JSON de cette valeur :

  Example

```
"value": {
          "value": "(the new value"),
          "dataType": "(the JSON datatypenew value")
        }
```

Statistiques du plan de données Neptune et résumé du graphique APIs

Actions liées aux statistiques des graphes de propriétés :

- GetPropertygraphStatistics (action)
- ManagePropertygraphStatistics (action)
- DeletePropertygraphStatistics (action)
- GetPropertygraphSummary (action)

Structures des statistiques :

- Statistics (structure)

- StatisticsSummary (structure)
- DeleteStatisticsValueMap (structure)
- RefreshStatisticsIdMap (structure)
- NodeStructure (structure)
- EdgeStructure (structure)
- SubjectStructure (structure)
- PropertygraphSummaryValueMap (structure)
- PropertygraphSummary (structure)

GetPropertygraphStatistics (action)

Le nom de la AWS CLI pour cette API est :`get-propertygraph-statistics`.

Obtient les statistiques du graphe de propriétés (Gremlin et openCypher).

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action GetStatisticsStatusneptune-db : IAM dans ce cluster.

Demande

- Aucun paramètre de demande.

Réponse

- payload : obligatoire : objet Statistiques.

  Statistiques pour les données du graphe de propriétés.
- status : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Code de retour HTTP de la demande. Si la demande aboutit, le code est 200. Consultez la section Codes d'erreur courants pour les demandes de statistiques DFE pour obtenir une liste des erreurs courantes.

Erreurs

- BadRequestException

- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ManagePropertygraphStatistics (action)

Le nom de la AWS CLI pour cette API est :`manage-propertygraph-statistics`.

Gère la génération et l'utilisation des statistiques des graphes de propriétés.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action ManageStatisticsneptune-db : IAM dans ce cluster.

Demande

- mode(dans la CLI :`--mode`) — a StatisticsAutoGenerationMode, de type : `string` (une chaîne codée en UTF-8).

  Mode de génération des statistiques. Correspond à `DISABLE_AUTOCOMPUTE`, `ENABLE_AUTOCOMPUTE` ou `REFRESH` (le dernier mode déclenche manuellement la génération de statistiques DFE).

Réponse

- payload : objet RefreshStatisticsIdMap.

N'est renvoyé que pour le mode d'actualisation.

- status : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Code de retour HTTP de la demande. Si la demande aboutit, le code est 200.

Erreurs

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

DeletePropertygraphStatistics (action)

Le nom de la AWS CLI pour cette API est :`delete-propertygraph-statistics`.

Supprime les statistiques pour les données Gremlin et openCypher (graphe de propriétés).

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action DeleteStatisticsneptune-db : IAM dans ce cluster.

Demande

- Aucun paramètre de demande.

Réponse

- payload : objet DeleteStatisticsValueMap.

  Charge utile de la suppression.

- status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Statut d'annulation.

- statusCode : entier de type : `integer` (entier signé de 32 bits).

  Code de réponse HTTP 200 si la suppression a réussi, ou 204 s'il n'y avait aucune statistique à supprimer.

Erreurs

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

GetPropertygraphSummary (action)

Le nom de la AWS CLI pour cette API est :`get-propertygraph-summary`.

Génère un résumé de graphe de propriétés.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action GetGraphSummaryneptune-db : IAM dans ce cluster.

Demande

- mode(dans la CLI :`--mode`) — a GraphSummaryType, de type : `string` (une chaîne codée en UTF-8).

  Le mode peut prendre l'une des deux valeurs suivantes : `BASIC` (valeur par défaut) et `DETAILED`.

Réponse

- payload : objet PropertygraphSummaryValueMap.

  Charge utile contenant la réponse de résumé de graphe de propriétés.
- statusCode : entier de type : `integer` (entier signé de 32 bits).

  Code de retour HTTP de la demande. Si la demande aboutit, le code est 200.

Erreurs

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Structures des statistiques :

Statistics (structure)

Contient des informations statistiques. Le moteur DFE utilise les informations relatives aux données d'un graphe Neptune pour effectuer des compromis efficaces lors de la planification de l'exécution des requêtes. Ces informations prennent la forme de statistiques qui incluent ce que l'on appelle des ensembles de caractéristiques et des statistiques de prédicats qui contribuent à guider la planification des requêtes. Consultez Gestion des statistiques à utiliser par le DFE Neptune.

Champs

- active : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si la génération des statistiques est activée.
- autoCompute : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Indique si la génération automatique des statistiques est activée ou non.
- date— Il s'agit d'un SyntheticTimestamp _date_time, de type : `string` (une chaîne codée en UTF-8).

  Heure UTC à laquelle les statistiques DFE ont été générées pour la dernière fois.
- note : chaîne de type : `string` (chaîne encodée en UTF-8).

  Remarque concernant les problèmes liés à la non-validité des statistiques.
- signatureInfo : objet StatisticsSummary.

  Une StatisticsSummary structure qui contient :
  - `signatureCount` : nombre total de signatures pour tous les ensembles de caractéristiques.
  - `instanceCount` : nombre total d'instances d'ensembles de caractéristiques.
  - `predicateCount` : nombre total de prédicats uniques.
- statisticsId : chaîne de type : `string` (chaîne encodée en UTF-8).

  Indique l'ID de la génération de statistiques en cours d'exécution. La valeur -1 indique qu'aucune statistique n'a été générée.

StatisticsSummary (structure)

Informations sur les ensembles de caractéristiques générés dans les statistiques.

Champs

- instanceCount : entier de type : `integer` (entier signé de 32 bits).

  Nombre total d'instances d'ensembles de caractéristiques.
- predicateCount : entier de type : `integer` (entier signé de 32 bits).

  Nombre total de prédicats uniques.
- signatureCount : entier de type : `integer` (entier signé de 32 bits).

  Nombre total de signatures pour tous les ensembles de caractéristiques.

DeleteStatisticsValueMap (structure)

La charge utile pour. DeleteStatistics

Champs

- active : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Statut actuel des statistiques.
- statisticsId : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID d'exécution de la génération de statistiques en cours.

RefreshStatisticsIdMap (structure)

Statistiques pour le mode `REFRESH`.

Champs

- statisticsId : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID d'exécution de la génération de statistiques en cours.

NodeStructure (structure)

Structure d'un nœud.

Champs

- count : long de type : `long` (entier signé de 64 bits).

  Nombre de nœuds dotés de cette structure spécifique.
- distinctOutgoingEdgeLabels : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des différentes étiquettes d'arête sortantes présentes dans cette structure spécifique.
- nodeProperties : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des propriétés de nœud présentes dans cette structure spécifique.

EdgeStructure (structure)

Structure d'une arête.

Champs

- count : long de type : `long` (entier signé de 64 bits).

  Nombre d'arêtes dotées de cette structure spécifique.
- edgeProperties : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des propriétés d'arête présentes dans cette structure spécifique.

SubjectStructure (structure)

Structure de sujet.

Champs

- count : long de type : `long` (entier signé de 64 bits).

  Nombre d'occurrences de cette structure spécifique.
- predicates : chaîne de type : `string` (chaîne encodée en UTF-8).

Liste des prédicats présents dans cette structure spécifique.

PropertygraphSummaryValueMap (structure)

Charge utile de la réponse de résumé de graphe de propriétés.

Champs

- graphSummary : objet PropertygraphSummary.

  Résumé du graphe.
- lastStatisticsComputationTime— Il s'agit d'un SyntheticTimestamp _date_time, de type : `string` (une chaîne codée en UTF-8).

  Horodatage, au format ISO 8601, indiquant l'heure à laquelle Neptune a calculé ses statistiques pour la dernière fois.
- version : chaîne de type : `string` (chaîne encodée en UTF-8).

  Version de la réponse de résumé de graphe.

PropertygraphSummary (structure)

L'API de résumé de graphe renvoie une liste en lecture seule des étiquettes et des clés de propriété des nœuds et des arêtes, ainsi que le nombre de nœuds, d'arêtes et de propriétés. Consultez Réponse de résumé pour un graphe de propriétés (PG).

Champs

- edgeLabels : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des étiquettes d'arête distinctes dans le graphe.
- edgeProperties— Ce sont LongValuedMap des objets. Il s'agit d'un tableau cartographique de paires clé-valeur où :

  Chaque clé est une chaîne de type : `string` (chaîne encodée en UTF-8).

  Chaque valeur est un long de type : `long` (entier signé de 64 bits).

Liste des propriétés d'arêtes distinctes dans le graphe et nombre d'arêtes où chaque propriété est utilisée.

- edgeStructures : tableau d'objets EdgeStructure.

  Ce champ n'est présent que lorsque le mode demandé est `DETAILED`. Contient une liste des structures d'arête.

- nodeLabels : chaîne de type : `string` (chaîne encodée en UTF-8).

  Liste des étiquettes de nœud distinctes dans le graphe.

- nodeProperties— Ce sont LongValuedMap des objets. Il s'agit d'un tableau cartographique de paires clé-valeur où :

  Chaque clé est une chaîne de type : `string` (chaîne encodée en UTF-8).

  Chaque valeur est un long de type : `long` (entier signé de 64 bits).

  Nombre de propriétés de nœuds distinctes dans le graphe.

- nodeStructures : tableau d'objets NodeStructure.

  Ce champ n'est présent que lorsque le mode demandé est `DETAILED`. Contient une liste des structures de nœud.

- numEdgeLabels : long de type : `long` (entier signé de 64 bits).

  Nombre d'étiquettes d'arête distinctes dans le graphe.

- numEdgeProperties : long de type : `long` (entier signé de 64 bits).

  Nombre de propriétés d'arêtes distinctes dans le graphe.

- numEdges : long de type : `long` (entier signé de 64 bits).

  Nombre d'arêtes dans le graphe.

- numNodeLabels : long de type : `long` (entier signé de 64 bits).

  Nombre d'étiquettes de nœuds distinctes dans le graphe.

- numNodeProperties : long de type : `long` (entier signé de 64 bits).

  Liste des propriétés de nœud distinctes dans le graphe et nombre de nœuds où chaque propriété est utilisée.

- numNodes : long de type : `long` (entier signé de 64 bits).

  Nombre de nœuds dans le graphe.

- totalEdgePropertyValues : long de type : `long` (entier signé de 64 bits).

  Nombre total d'utilisations de toutes les propriétés d'arête.

- totalNodePropertyValues : long de type : `long` (entier signé de 64 bits).

  Nombre total d'utilisations de toutes les propriétés de nœud.

API de traitement de données Neptune ML

Actions de traitement des données :

- Démarrer MLData ProcessingJob (action)
- Liste MLData ProcessingJobs (action)
- Obtenir MLData ProcessingJob (action)
- Annuler MLData ProcessingJob (action)

Structures à usage général du ML :

- MlResourceDefinition (structure)
- MlConfigDefinition (structure)

Démarrer MLData ProcessingJob (action)

> Le nom de la AWS CLI pour cette API est :`start-ml-data-processing-job`.

Crée une tâche de traitement de données Neptune ML pour traiter les données de graphe exportées à partir de Neptune à des fins d'entraînement. Consultez la section Commande `dataprocessing`.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLModel DataProcessingJob Neptune-DB:Start IAM dans ce cluster.

Demande

- configFileName (dans la CLI : `--config-file-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Fichier de spécification de données qui explique comment charger les données de graphe exportées à des fins d'entraînement. Ce fichier est automatiquement généré par le kit d'exportation Neptune. L'argument par défaut est `training-data-configuration.json`.

- id (dans la CLI : `--id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de la nouvelle tâche. La valeur par défaut est un UUID généré automatiquement.

- inputDataS3Location (dans la CLI : `--input-data-s3-location`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'URI de l'emplacement Amazon S3 où vous souhaitez SageMaker télécharger les données nécessaires à l'exécution de la tâche de traitement des données.

- modelType (dans la CLI : `--model-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'un des deux types de modèles actuellement pris en charge par Neptune ML : modèles de graphes hétérogènes (`heterogeneous`) et graphes de connaissances (`kge`). La valeur par défaut est « aucun ». Si cette valeur n'est pas spécifiée, Neptune ML choisit automatiquement le type de modèle en fonction des données.

- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le nom de ressource Amazon (ARN) d'un rôle IAM qui SageMaker peut assumer l'exécution de tâches en votre nom. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

- previousDataProcessingJobId (dans la CLI : `--previous-data-processing-job-id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID d'une tâche de traitement des données terminée exécutée sur une version antérieure des données.

- processedDataS3Location (dans la CLI : `--processed-data-s3-location`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'URI de l'emplacement Amazon S3 où vous souhaitez SageMaker enregistrer les résultats d'une tâche de traitement de données.

- processingInstanceType (dans la CLI : `--processing-instance-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type d'instance ML utilisée lors du traitement des données. Sa mémoire doit être suffisamment grande pour contenir le jeu de données traité. La valeur par défaut est le plus petit type d'instance ml.r5 dont la mémoire est dix fois supérieure à la taille des données de graphe exportées sur le disque.

- processingInstanceVolumeSizeInGB (dans la CLI : `--processing-instance-volume-size-in-gb`) : entier de type : `integer` (entier signé de 32 bits).

  Taille du volume de disque de l'instance de traitement. Les données d'entrée et les données traitées étant stockées sur disque, la taille du volume doit être suffisamment grande pour contenir les deux jeux de données. La valeur par défaut est 0. Si ce paramètre n'est pas spécifié ou s'il est égal à 0, Neptune ML choisit automatiquement la taille du volume en fonction de la taille des données.

- processingTimeOutInSeconds (dans la CLI : `--processing-time-out-in-seconds`) : entier de type : `integer` (entier signé de 32 bits).

  Délai d'expiration en secondes de la tâche de traitement des données. La valeur par défaut de 86 400 (1 jour).

- s3OutputEncryptionKMSKey (dans la CLI : `--s-3-output-encryption-kms-key`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  La clé Amazon Key Management Service (Amazon KMS) SageMaker utilisée pour chiffrer le résultat de la tâche de traitement. La valeur par défaut est Aucun.

- sagemakerIamRoleArn (dans la CLI : `--sagemaker-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM à SageMaker exécuter. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

- securityGroupIds (dans la CLI : `--security-group-ids`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le groupe de sécurité VPC. IDs La valeur par défaut est Aucun.

- subnets (dans la CLI : `--subnets`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Les sous-réseaux IDs du VPC Neptune. La valeur par défaut est Aucun.

- volumeEncryptionKMSKey (dans la CLI : `--volume-encryption-kms-key`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  La clé Amazon Key Management Service (Amazon KMS) SageMaker utilisée pour chiffrer les données sur le volume de stockage attaché aux instances de calcul ML qui exécutent la tâche de formation. La valeur par défaut est Aucun.

Réponse

- arn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN de la tâche de traitement des données.
- creationTimeInMillis : long de type : `long` (entier signé de 64 bits).

  Temps nécessaire à la création de la tâche de traitement, en millisecondes.
- id : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID unique de la nouvelle tâche de traitement de données.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Liste MLData ProcessingJobs (action)

Le nom de la AWS CLI pour cette API est :`list-ml-data-processing-jobs`.

Renvoie une liste des tâches de traitement de données Neptune ML. Consultez Liste des tâches de traitement de données actives à l'aide de la commande dataprocessing Neptune ML.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLData ProcessingJobs Neptune-DB:list IAM dans ce cluster.

Demande

- maxItems(dans la CLI :`--max-items`) — une liste ListMLDataProcessingJobsInputMaxItemsInteger, de type : `integer` (un entier signé de 32 bits), pas moins de 1 ou plus de 1024 ? 1er ? s.

  Nombre maximum d'éléments à récupérer (allant de 1 à 1 024, avec une valeur par défaut de 10).
- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM qui fournit à Neptune l'accès aux ressources Amazon S3 et SageMaker à celles-ci. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

Réponse

- ids : chaîne de type : `string` (chaîne encodée en UTF-8).

  Une page répertoriant les tâches de traitement de données IDs.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException

- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Obtenir MLData ProcessingJob (action)

Le nom de la AWS CLI pour cette API est :`get-ml-data-processing-job`.

Extrait des informations sur une tâche de traitement de données spécifiée. Consultez la section Commande `dataprocessing`.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLData ProcessingJobStatus Neptune-DB:Neptune-DB:get IAM dans ce cluster.

Demande

- id (dans la CLI : `--id`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de la tâche de traitement des données à récupérer.
- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM qui fournit à Neptune l'accès aux ressources Amazon S3 et SageMaker à celles-ci. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

Réponse

- id : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de cette tâche de traitement de données.
- processingJob : objet MlResourceDefinition.

  Définition de la tâche de traitement des données.

- status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Statut de la tâche de traitement des données.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Annuler MLData ProcessingJob (action)

Le nom de la AWS CLI pour cette API est :`cancel-ml-data-processing-job`.

Annule une tâche de traitement de données Neptune ML. Consultez la section Commande `dataprocessing`.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLData ProcessingJob Neptune-DB:Cancel IAM dans ce cluster.

Demande

- clean (dans la CLI : `--clean`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  S'il est défini sur `TRUE`, cet indicateur indique que tous les artefacts Neptune ML S3 doivent être supprimés lorsque la tâche est arrêtée. L'argument par défaut est `FALSE`.

- id (dans la CLI : `--id`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de la tâche de traitement de données.
- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM qui fournit à Neptune l'accès aux ressources Amazon S3 et SageMaker à celles-ci. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

Réponse

- status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Statut de la demande d'annulation.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Structures à usage général du ML :

MlResourceDefinition (structure)

Définit une ressource Neptune ML.

Champs

- arn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN de la ressource.
- cloudwatchLogUrl : chaîne de type : `string` (chaîne encodée en UTF-8).

  URL du CloudWatch journal de la ressource.
- failureReason : chaîne de type : `string` (chaîne encodée en UTF-8).

  Raison de l'échec, en cas d'échec.
- name : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la ressource.
- outputLocation : chaîne de type : `string` (chaîne encodée en UTF-8).

  Emplacement de la sortie.
- status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Statut de la ressource.

MlConfigDefinition (structure)

Contient une configuration Neptune ML.

Champs

- arn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN de la configuration.
- name : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom de la configuration

API d'entraînement de modèle Neptune ML

Actions d'entraînement de modèle :

- Démarrer MLModel TrainingJob (action)

- Liste MLModel TrainingJobs (action)
- Obtenir MLModel TrainingJob (action)
- Annuler MLModel TrainingJob (action)

Structures d'entraînement des modèles :

- CustomModelTrainingParameters (structure)

Démarrer MLModel TrainingJob (action)

Le nom de la AWS CLI pour cette API est :`start-ml-model-training-job`.

Crée une tâche d'entraînement de modèle Neptune ML. Consultez Entraînement des modèles à l'aide de la commande `modeltraining`.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLModel TrainingJob Neptune-DB:Start IAM dans ce cluster.

Demande

- baseProcessingInstanceType (dans la CLI : `--base-processing-instance-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type d'instance ML utilisé pour préparer et gérer l'entraînement des modèles ML. Il s'agit d'une instance de CPU choisie en fonction des besoins en mémoire pour le traitement des données et du modèle d'entraînement.
- customModelTrainingParameters (dans la CLI : `--custom-model-training-parameters`) : objet CustomModelTrainingParameters.

  Configuration pour l'entraînement d'un modèle personnalisé. Il s'agit d'un objet JSON.
- dataProcessingJobId (dans la CLI : `--data-processing-job-id`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la tâche de traitement des données terminée qui a créé les données avec lesquelles l'entraînement fonctionnera.
- enableManagedSpotTraining (dans la CLI : `--enable-managed-spot-training`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

Optimise le coût d'entraînement des modèles de machine learning à l'aide d'instances Spot Amazon Elastic Compute Cloud. L'argument par défaut est `False`.

- id (dans la CLI : `--id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de la nouvelle tâche. La valeur par défaut est un UUID généré automatiquement.

- maxHPONumberOfTrainingJobs (dans la CLI : `--max-hpo-number-of-training-jobs`) : entier de type : `integer` (entier signé de 32 bits).

  Nombre total maximal de tâches d'entraînement à démarrer pour la tâche de réglage des hyperparamètres. La valeur par défaut est 2. Neptune ML ajuste automatiquement les hyperparamètres du modèle de machine learning. Pour obtenir un modèle performant, utilisez au moins 10 tâches (en d'autres termes, définissez la valeur `maxHPONumberOfTrainingJobs` sur 10). En général, plus le réglage est long, meilleurs sont les résultats.

- maxHPOParallelTrainingJobs (dans la CLI : `--max-hpo-parallel-training-jobs`) : entier de type : `integer` (entier signé de 32 bits).

  Nombre maximal de tâches d'entraînement à démarrer pour la tâche de réglage des hyperparamètres. La valeur par défaut est 2. Le nombre de tâches parallèles que vous pouvez exécuter est limité par les ressources disponibles sur l'instance d'entraînement.

- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM qui fournit à Neptune l'accès aux ressources Amazon S3 et SageMaker à celles-ci. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

- previousModelTrainingJobId (dans la CLI : `--previous-model-training-job-id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID d'une tâche d'entraînement de modèle terminée que vous souhaitez actualiser progressivement en fonction des données mises à jour.

- s3OutputEncryptionKMSKey (dans la CLI : `--s-3-output-encryption-kms-key`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  La clé Amazon Key Management Service (KMS) SageMaker utilisée pour chiffrer le résultat de la tâche de traitement. La valeur par défaut est Aucun.

- sagemakerIamRoleArn (dans la CLI : `--sagemaker-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM à SageMaker exécuter. Il doit être répertorié dans le groupe de paramètres de votre cluster de base de données, sinon une erreur se produira.

- securityGroupIds (dans la CLI : `--security-group-ids`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le groupe de sécurité VPC. IDs La valeur par défaut est Aucun.

- subnets (dans la CLI : `--subnets`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Les sous-réseaux IDs du VPC Neptune. La valeur par défaut est Aucun.

- trainingInstanceType (dans la CLI : `--training-instance-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type d'instance ML utilisé pour l'entraînement des modèles. Tous les modèles Neptune ML prennent en charge l'entraînement de CPU, de GPU et de multiGPU. L'argument par défaut est `ml.p3.2xlarge`. Le choix du type d'instance approprié pour l'entraînement dépend du type de tâche, de la taille du graphe et de votre budget.

- trainingInstanceVolumeSizeInGB (dans la CLI : `--training-instance-volume-size-in-gb`) : entier de type : `integer` (entier signé de 32 bits).

  Taille du volume de disque de l'instance d'entraînement. Les données d'entrée et le modèle de sortie étant stockés sur disque, la taille du volume doit être suffisamment grande pour contenir les deux jeux de données. La valeur par défaut est 0. Si elle n'est pas spécifiée ou si elle est égale à 0, Neptune ML sélectionne une taille de volume de disque en fonction de la recommandation générée lors de l'étape de traitement des données.

- trainingTimeOutInSeconds (dans la CLI : `--training-time-out-in-seconds`) : entier de type : `integer` (entier signé de 32 bits).

  Délai d'attente en secondes de la tâche d'entraînement. La valeur par défaut de 86 400 (1 jour).

- trainModelS3Location (dans la CLI : `--train-model-s3-location`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Emplacement dans Amazon S3 où les artefacts de modèle doivent être stockés.

- volumeEncryptionKMSKey (dans la CLI : `--volume-encryption-kms-key`) : chaîne de type : `string` (chaîne encodée en UTF-8).

La clé Amazon Key Management Service (KMS) SageMaker utilisée pour chiffrer les données sur le volume de stockage attaché aux instances de calcul ML qui exécutent la tâche de formation. La valeur par défaut est Aucun.

Réponse

- arn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN de la nouvelle tâche d'entraînement du modèle.
- creationTimeInMillis : long de type : `long` (entier signé de 64 bits).

  Temps de création de la tâche d'entraînement du modèle, en millisecondes.
- id : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID unique de la nouvelle tâche d'entraînement du modèle.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Liste MLModel TrainingJobs (action)

Le nom de la AWS CLI pour cette API est :`list-ml-model-training-jobs`.

Répertorie les tâches d'entraînement de modèle Neptune ML. Consultez Entraînement des modèles à l'aide de la commande `modeltraining`.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLModel TrainingJobs Neptune-db:neptune-db:list IAM dans ce cluster.

Demande

- maxItems(dans la CLI :`--max-items`) — une liste MLModelTrainingJobsInputMaxItemsInteger, de type : `integer` (un entier signé de 32 bits), pas moins de 1 ou plus de 1024 ? 1er ? s.

  Nombre maximum d'éléments à récupérer (allant de 1 à 1 024, avec une valeur par défaut de 10).
- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM qui fournit à Neptune l'accès aux ressources Amazon S3 et SageMaker à celles-ci. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

Réponse

- ids : chaîne de type : `string` (chaîne encodée en UTF-8).

  Une page de la liste des emplois de formation modèles IDs.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException

- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Obtenir MLModel TrainingJob (action)

Le nom de la AWS CLI pour cette API est :`get-ml-model-training-job`.

Récupère des informations sur une tâche d'entraînement de modèle Neptune ML. Consultez Entraînement des modèles à l'aide de la commande `modeltraining`.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLModel TrainingJobStatus Neptune-DB:Get IAM dans ce cluster.

Demande

- id (dans la CLI : `--id`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de la tâche d'entraînement de modèle à récupérer.
- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM qui fournit à Neptune l'accès aux ressources Amazon S3 et SageMaker à celles-ci. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

Réponse

- hpoJob : objet MlResourceDefinition.

  Tâche HPO.
- id : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de cette tâche d'entraînement de modèle.
- mlModels : tableau d'objets MlConfigDefinition.

  Liste des configurations des modèles ML utilisés.

- modelTransformJob : objet MlResourceDefinition.

  Tâche de transformation de modèle.

- processingJob : objet MlResourceDefinition.

  Tâche de traitement des données.

- status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Statut de la nouvelle tâche d'entraînement du modèle.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Annuler MLModel TrainingJob (action)

Le nom de la AWS CLI pour cette API est :`cancel-ml-model-training-job`.

Annule une tâche d'entraînement de modèle Neptune ML. Consultez Entraînement des modèles à l'aide de la commande `modeltraining`.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLModel TrainingJob Neptune-DB:Cancel IAM dans ce cluster.

Demande

- clean (dans la CLI : `--clean`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  S'il est défini sur TRUE, cet indicateur indique que tous les artefacts Amazon S3 doivent être supprimés lorsque la tâche est arrêtée. L'argument par défaut est `FALSE`.
- id (dans la CLI : `--id`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de la tâche d'entraînement de modèle à annuler.
- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM qui fournit à Neptune l'accès aux ressources Amazon S3 et SageMaker à celles-ci. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

Réponse

- status : chaîne de type : `string` (chaîne encodée en UTF-8).

  État de l'annulation.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Structures d'entraînement des modèles :

CustomModelTrainingParameters (structure)

Contient les paramètres d'entraînement personnalisés des modèles. Consultez Modèles personnalisés dans Neptune ML.

Champs

- sourceS3DirectoryPath : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Chemin d'accès vers l'emplacement Amazon S3 où se trouve le module Python implémentant votre modèle. Il doit pointer vers un emplacement Amazon S3 existant valide contenant, au minimum, un script d'entraînement, un script de transformation et un fichier `model-hpo-configuration.json`.
- trainingEntryPointScript : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du point d'entrée dans votre module d'un script qui effectue l'entraînement du modèle et utilise des hyperparamètres comme arguments de ligne de commande, y compris des hyperparamètres fixes. L'argument par défaut est `training.py`.
- transformEntryPointScript : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du point d'entrée dans le module d'un script qui doit être exécuté une fois que le modèle le plus approprié issu de la recherche par hyperparamètres a été identifié, afin de calculer les artefacts nécessaires au déploiement du modèle. Il devrait pouvoir s'exécuter sans arguments de ligne de commande. La valeur par défaut est `transform.py`.

API de transformation de modèle Neptune ML

Actions de transformation de modèle :

- Démarrer MLModel TransformJob (action)
- Liste MLModel TransformJobs (action)
- Obtenir MLModel TransformJob (action)
- Annuler MLModel TransformJob (action)

Structures de transformation de modèle :

- CustomModelTransformParameters (structure)

Démarrer MLModel TransformJob (action)

Le nom de la AWS CLI pour cette API est :`start-ml-model-transform-job`.

Crée une tâche de transformation de modèle. Consultez Utilisation d'un modèle entraîné pour générer de nouveaux artefacts de modèle.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLModel TransformJob Neptune-DB:Start IAM dans ce cluster.

Demande

- baseProcessingInstanceType (dans la CLI : `--base-processing-instance-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type d'instance ML utilisé pour préparer et gérer l'entraînement des modèles ML. Il s'agit d'une instance de calcul ML choisie en fonction des besoins en mémoire pour le traitement des données et du modèle d'entraînement.
- baseProcessingInstanceVolumeSizeInGB (dans la CLI : `--base-processing-instance-volume-size-in-gb`) : entier de type : `integer` (entier signé de 32 bits).

  Taille du volume de disque de l'instance d'entraînement, en gigaoctets. La valeur par défaut est 0. Les données d'entrée et le modèle de sortie étant stockés sur disque, la taille du volume doit être suffisamment grande pour contenir les deux jeux de données. Si elle n'est pas spécifiée ou si elle est égale à 0, Neptune ML sélectionne une taille de volume de disque en fonction de la recommandation générée lors de l'étape de traitement des données.
- customModelTransformParameters (dans la CLI : `--custom-model-transform-parameters`) : objet CustomModelTransformParameters.

  Informations de configuration d'une transformation de modèle à l'aide d'un modèle personnalisé. L'objet `customModelTransformParameters` contient les champs suivants, dont les valeurs doivent être compatibles avec les paramètres de modèle enregistrés lors de la tâche d'entraînement :
- dataProcessingJobId (dans la CLI : `--data-processing-job-id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

ID d'une tâche de traitement des données terminée. Vous devez inclure `dataProcessingJobId` et `mlModelTrainingJobId` ou `trainingJobName`.

- id (dans la CLI : `--id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de la nouvelle tâche. La valeur par défaut est un UUID généré automatiquement.

- mlModelTrainingJobId (dans la CLI : `--ml-model-training-job-id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID d'une tâche d'entraînement de modèle terminée. Vous devez inclure `dataProcessingJobId` et `mlModelTrainingJobId` ou `trainingJobName`.

- modelTransformOutputS3Location (dans la CLI : `--model-transform-output-s3-location`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Emplacement dans Amazon S3 où les artefacts de modèle doivent être stockés.

- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM qui fournit à Neptune l'accès aux ressources Amazon S3 et SageMaker à celles-ci. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

- s3OutputEncryptionKMSKey (dans la CLI : `--s-3-output-encryption-kms-key`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  La clé Amazon Key Management Service (KMS) SageMaker utilisée pour chiffrer le résultat de la tâche de traitement. La valeur par défaut est Aucun.

- sagemakerIamRoleArn (dans la CLI : `--sagemaker-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM à SageMaker exécuter. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

- securityGroupIds (dans la CLI : `--security-group-ids`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le groupe de sécurité VPC. IDs La valeur par défaut est Aucun.

- subnets (dans la CLI : `--subnets`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Les sous-réseaux IDs du VPC Neptune. La valeur par défaut est Aucun.

- trainingJobName (dans la CLI : `--training-job-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Le nom d'une tâche de SageMaker formation terminée. Vous devez inclure `dataProcessingJobId` et `mlModelTrainingJobId` ou `trainingJobName`.

- volumeEncryptionKMSKey (dans la CLI : `--volume-encryption-kms-key`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  La clé Amazon Key Management Service (KMS) SageMaker utilisée pour chiffrer les données sur le volume de stockage attaché aux instances de calcul ML qui exécutent la tâche de formation. La valeur par défaut est Aucun.

Réponse

- arn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN de la tâche de transformation du modèle.

- creationTimeInMillis : long de type : `long` (entier signé de 64 bits).

  Temps de création de la tâche de transformation du modèle, en millisecondes.

- id : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID unique de la nouvelle tâche de transformation du modèle.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

- IllegalArgumentException
- TooManyRequestsException

Liste MLModel TransformJobs (action)

Le nom de la AWS CLI pour cette API est :`list-ml-model-transform-jobs`.

Renvoie une liste des tâches de transformation du modèle IDs. Consultez Utilisation d'un modèle entraîné pour générer de nouveaux artefacts de modèle.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLModel TransformJobs Neptune-DB:list IAM dans ce cluster.

Demande

- maxItems(dans la CLI :`--max-items`) — une liste ListMLModelTransformJobsInputMaxItemsInteger, de type : `integer` (un entier signé de 32 bits), pas moins de 1 ou plus de 1024 ? 1er ? s.

  Nombre maximum d'éléments à récupérer (allant de 1 à 1 024, avec une valeur par défaut de 10).
- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM qui fournit à Neptune l'accès aux ressources Amazon S3 et SageMaker à celles-ci. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

Réponse

- ids : chaîne de type : `string` (chaîne encodée en UTF-8).

  Une page de la liste des transformations du modèle IDs.

Erreurs

- UnsupportedOperationException
- BadRequestException

- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Obtenir MLModel TransformJob (action)

Le nom de la AWS CLI pour cette API est :`get-ml-model-transform-job`.

Génère des informations sur une tâche de transformation de modèle spécifiée. Consultez Utilisation d'un modèle entraîné pour générer de nouveaux artefacts de modèle.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLModel TransformJobStatus Neptune-DB:Get IAM dans ce cluster.

Demande

- id (dans la CLI : `--id`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de la tâche de transformation de modèle à récupérer.
- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM qui fournit à Neptune l'accès aux ressources Amazon S3 et SageMaker à celles-ci. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

Réponse

- baseProcessingJob : objet MlResourceDefinition.

Tâche de traitement des données de base.

- id : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique de la tâche de transformation de modèle à récupérer.

- models : tableau d'objets MlConfigDefinition.

  Liste des informations de configuration pour les modèles utilisés.

- remoteModelTransformJob : objet MlResourceDefinition.

  Tâche de transformation de modèle à distance.

- status : chaîne de type : `string` (chaîne encodée en UTF-8).

  Statut de la tâche de transformation du modèle.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Annuler MLModel TransformJob (action)

Le nom de la AWS CLI pour cette API est :`cancel-ml-model-transform-job`.

Annule une tâche de transformation de modèle spécifiée. Consultez Utilisation d'un modèle entraîné pour générer de nouveaux artefacts de modèle.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLModel TransformJob Neptune-DB:Cancel IAM dans ce cluster.

Demande

- clean (dans la CLI : `--clean`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Si cet indicateur est défini sur `TRUE`, tous les artefacts de Neptune ML S3 doivent être supprimés à l'arrêt de la tâche. L'argument par défaut est `FALSE`.
- id (dans la CLI : `--id`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID unique de la tâche de transformation du modèle à annuler.
- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM qui fournit à Neptune l'accès aux ressources Amazon S3 et SageMaker à celles-ci. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

Réponse

- status : chaîne de type : `string` (chaîne encodée en UTF-8).

  État de l'annulation.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException

- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Structures de transformation de modèle :

CustomModelTransformParameters (structure)

Contient les paramètres de transformation personnalisés des modèles. Consultez Utilisation d'un modèle entraîné pour générer de nouveaux artefacts de modèle.

Champs

- sourceS3DirectoryPath : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Chemin d'accès vers l'emplacement Amazon S3 où se trouve le module Python implémentant votre modèle. Il doit pointer vers un emplacement Amazon S3 existant valide contenant, au minimum, un script d'entraînement, un script de transformation et un fichier `model-hpo-configuration.json`.
- transformEntryPointScript : chaîne de type : `string` (chaîne encodée en UTF-8).

  Nom du point d'entrée dans le module d'un script qui doit être exécuté une fois que le modèle le plus approprié issu de la recherche par hyperparamètres a été identifié, afin de calculer les artefacts nécessaires au déploiement du modèle. Il devrait pouvoir s'exécuter sans arguments de ligne de commande. L'argument par défaut est `transform.py`.

API de point de terminaison d'inférence Neptune ML

Actions du point de terminaison d'inférence :

- Créer MLEndpoint (action)
- Liste MLEndpoints (action)
- Obtenir MLEndpoint (action)
- Supprimer MLEndpoint (action)

Créer MLEndpoint (action)

Le nom de la AWS CLI pour cette API est :`create-ml-endpoint`.

Crée un point de terminaison d'inférence Neptune ML qui vous permet d'interroger un modèle spécifique construit par le processus d'entraînement des modèles. Consultez Gestion des points de terminaison d'inférence à l'aide de la commande endpoints.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLEndpoint Neptune-DB:Create IAM dans ce cluster.

Demande

- id (dans la CLI : `--id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique du nouveau point de terminaison d'inférence. Le nom par défaut est un nom horodaté généré automatiquement.
- instanceCount (dans la CLI : `--instance-count`) : entier de type : `integer` (entier signé de 32 bits).

  Le nombre minimum d' EC2 instances Amazon à déployer sur un point de terminaison à des fins de prévision. La valeur par défaut est 1.
- instanceType (dans la CLI : `--instance-type`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type d'instance Neptune ML à utiliser pour la maintenance en ligne. L'argument par défaut est `ml.m5.xlarge`. Le choix de l'instance ML pour un point de terminaison d'inférence dépend du type de tâche, de la taille du graphe et de votre budget.
- mlModelTrainingJobId (dans la CLI : `--ml-model-training-job-id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la tâche d'entraînement de modèle terminée qui a créé le modèle vers lequel le point final d'inférence pointera. Vous devez fournir `mlModelTrainingJobId` ou `mlModelTransformJobId`.
- mlModelTransformJobId (dans la CLI : `--ml-model-transform-job-id`) : chaîne de type : `string` (chaîne encodée en UTF-8).

ID de la tâche de transformation de modèle terminée. Vous devez fournir `mlModelTrainingJobId` ou `mlModelTransformJobId`.

- modelName (dans la CLI : `--model-name`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  Type de modèle pour l'entraînement. Par défaut, le modèle Neptune ML est automatiquement basé sur le `modelType` utilisé dans le traitement des données, mais vous pouvez spécifier ici un autre type de modèle. La valeur par défaut est `rgcn` pour les graphes hétérogènes et `kge` pour les graphes de connaissances. La seule valeur valide pour les graphes hétérogènes est `rgcn`. Les valeurs valides pour les graphes de connaissances sont `kge`, `transe`, `distmult` et `rotate`.
- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM fournissant à Neptune l'accès aux ressources Amazon S3 SageMaker et à celles-ci. Cela doit être indiqué dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.
- update (dans la CLI : `--update`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Si ce paramètre est défini sur `true`, `update` indique qu'il s'agit d'une demande de mise à jour. L'argument par défaut est `false`. Vous devez fournir `mlModelTrainingJobId` ou `mlModelTransformJobId`.
- volumeEncryptionKMSKey (dans la CLI : `--volume-encryption-kms-key`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  La clé Amazon Key Management Service (Amazon KMS) SageMaker utilisée pour chiffrer les données sur le volume de stockage attaché aux instances de calcul ML qui exécutent la tâche de formation. La valeur par défaut est Aucun.

Réponse

- arn : chaîne de type : `string` (chaîne encodée en UTF-8).

  ARN du nouveau point de terminaison d'inférence.
- creationTimeInMillis : long de type : `long` (entier signé de 64 bits).

  Heure de création du point de terminaison, en millisecondes.

- id : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique du point de terminaison d'inférence.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Liste MLEndpoints (action)

Le nom de la AWS CLI pour cette API est :`list-ml-endpoints`.

Répertorie les points de terminaison d'inférence existants. Consultez Gestion des points de terminaison d'inférence à l'aide de la commande endpoints.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLEndpoints Neptune-DB:list IAM dans ce cluster.

Demande

- maxItems(dans la CLI :`--max-items`) — une liste MLEndpointsInputMaxItemsInteger, de type : `integer` (un entier signé de 32 bits), pas moins de 1 ou plus de 1024 ? 1er ? s.

  Nombre maximum d'éléments à renvoyer (compris entre 1 et 1 024, avec une valeur par défaut de 10).

- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM qui fournit à Neptune l'accès aux ressources Amazon S3 et SageMaker à celles-ci. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

Réponse

- ids : chaîne de type : `string` (chaîne encodée en UTF-8).

  Une page de la liste des points de terminaison IDs d'inférence.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Obtenir MLEndpoint (action)

Le nom de la AWS CLI pour cette API est :`get-ml-endpoint`.

Récupère les détails d'un point de terminaison d'inférence. Consultez Gestion des points de terminaison d'inférence à l'aide de la commande endpoints.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLEndpointNeptune-DB:Get Status IAM dans ce cluster.

Demande

- id (dans la CLI : `--id`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique du point de terminaison d'inférence.
- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM qui fournit à Neptune l'accès aux ressources Amazon S3 et SageMaker à celles-ci. Cela doit être répertorié dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

Réponse

- endpoint : objet MlResourceDefinition.

  Définition du point de terminaison.
- endpointConfig : objet MlConfigDefinition.

  Configuration du point de terminaison.
- id : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique du point de terminaison d'inférence.
- status : chaîne de type : `string` (chaîne encodée en UTF-8).

  État du point de terminaison d'inférence.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException

- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Supprimer MLEndpoint (action)

Le nom de la AWS CLI pour cette API est :`delete-ml-endpoint`.

Annule la création d'un point de terminaison d'inférence Neptune ML. Consultez Gestion des points de terminaison d'inférence à l'aide de la commande endpoints.

Lorsque vous appelez cette opération dans un cluster Neptune sur lequel l'authentification IAM est activée, l'utilisateur ou le rôle IAM à l'origine de la demande doit être associé à une politique autorisant l'action MLEndpoint Neptune-DB:Delete IAM dans ce cluster.

Demande

- clean (dans la CLI : `--clean`) : valeur booléenne de type : `boolean` (valeur booléenne : true ou false).

  Si cet indicateur est défini sur `TRUE`, tous les artefacts de Neptune ML S3 doivent être supprimés à l'arrêt de la tâche. L'argument par défaut est `FALSE`.
- id (dans la CLI : `--id`) : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Identifiant unique du point de terminaison d'inférence.
- neptuneIamRoleArn (dans la CLI : `--neptune-iam-role-arn`) : chaîne de type : `string` (chaîne encodée en UTF-8).

  L'ARN d'un rôle IAM fournissant à Neptune l'accès aux ressources Amazon S3 SageMaker et à celles-ci. Cela doit être indiqué dans le groupe de paramètres de votre cluster de bases de données. Dans le cas contraire, une erreur se produira.

Réponse

- status : chaîne de type : `string` (chaîne encodée en UTF-8).

  État de l'annulation.

Erreurs

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Exceptions relatives à l'API du plan de données Neptune

Exceptions :

- AccessDeniedException (structure)
- BadRequestException (structure)
- BulkLoadIdNotFoundException (structure)
- CancelledByUserException (structure)
- ClientTimeoutException (structure)
- ConcurrentModificationException (structure)
- ConstraintViolationException (structure)
- ExpiredStreamException (structure)
- FailureByQueryException (structure)
- IllegalArgumentException (structure)
- InternalFailureException (structure)

- InvalidArgumentException (structure)
- InvalidNumericDataException (structure)
- InvalidParameterException (structure)
- LoadUrlAccessDeniedException (structure)
- MalformedQueryException (structure)
- MemoryLimitExceededException (structure)
- MethodNotAllowedException (structure)
- MissingParameterException (structure)
- MLResourceNotFoundException (structure)
- ParsingException (structure)
- PreconditionsFailedException (structure)
- QueryLimitExceededException (structure)
- QueryLimitException (structure)
- QueryTooLargeException (structure)
- ReadOnlyViolationException (structure)
- S3Exception (structure)
- ServerShutdownException (structure)
- StatisticsNotAvailableException (structure)
- StreamRecordsNotFoundException (structure)
- ThrottlingException (structure)
- TimeLimitExceededException (structure)
- TooManyRequestsException (structure)
- UnsupportedOperationException (structure)
- UnloadUrlAccessDeniedException (structure)

AccessDeniedException (structure)

Générée en cas d'échec d'authentification ou d'autorisation.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

Codes de statut HTTP renvoyé avec l'exception.

- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.

- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

BadRequestException (structure)

Générée lorsqu'une demande est soumise et qu'elle ne peut pas être traitée.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.

- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.

- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande incorrecte.

BulkLoadIdNotFoundException (structure)

Générée lorsqu'un ID de tâche de chargement en bloc spécifié est introuvable.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.

- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.

- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

ID de tâche de chargement en bloc introuvable.

CancelledByUserException (structure)

Générée lorsqu'un utilisateur annule une demande.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

ClientTimeoutException (structure)

Générée lorsqu'une demande a expiré côté client.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

ConcurrentModificationException (structure)

Générée lorsqu'une demande tente de modifier des données qui sont modifiées simultanément par un autre processus.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

ConstraintViolationException (structure)

Générée lorsqu'une valeur d'un champ de demande ne satisfaisait pas aux contraintes requises.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

ExpiredStreamException (structure)

Générée lorsqu'une demande tente d'accéder à un flux expiré.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.

- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

FailureByQueryException (structure)

Générée lorsqu'une demande échoue.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

IllegalArgumentException (structure)

Générée lorsqu'un argument d'une demande n'est pas pris en charge.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

InternalFailureException (structure)

Générée lorsque le traitement de la demande a échoué de façon inattendue.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

InvalidArgumentException (structure)

Générée lorsqu'un argument d'une demande possède une valeur non valide.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

InvalidNumericDataException (structure)

Générée lorsque des données numériques non valides sont détectées lors du traitement d'une demande.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

Message détaillé décrivant le problème.

- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

InvalidParameterException (structure)

Générée lorsqu'une valeur de paramètre n'est pas valide.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande qui inclut un paramètre non valide.

LoadUrlAccessDeniedException (structure)

Générée lorsque l'accès à une URL de chargement spécifiée est refusé.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

MalformedQueryException (structure)

Générée lorsqu'une requête soumise est incorrecte du point de vue syntaxique ou lorsqu'elle n'est pas approuvée par une étape de validation supplémentaire.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande de requête mal formée.

MemoryLimitExceededException (structure)

Générée lorsqu'une demande échoue en raison de ressources mémoire insuffisantes. Une nouvelle tentative peut avoir lieu.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande qui a échoué.

MethodNotAllowedException (structure)

Générée lorsque la méthode HTTP utilisée par une requête n'est pas prise en charge par le point de terminaison utilisé.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

MissingParameterException (structure)

Générée lorsqu'un paramètre obligatoire est manquant.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande dans laquelle il manque le paramètre.

MLResourceNotFoundException (structure)

Générée lorsqu'une ressource de machine learning spécifiée est introuvable.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.

- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

ParsingException (structure)

Générée en cas de problème d'analyse.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

PreconditionsFailedException (structure)

Générée lorsqu'une condition préalable au traitement d'une demande n'est pas satisfaite.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

QueryLimitExceededException (structure)

Générée lorsque le nombre de requêtes actives dépasse ce que le serveur peut traiter. La requête en question pourra faire l'objet d'une nouvelle tentative lorsque le système sera moins occupé.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande qui a dépassé la limite.

QueryLimitException (structure)

Générée lorsque la taille d'une requête dépasse la limite du système.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande qui a dépassé la limite.

QueryTooLargeException (structure)

Générée lorsque le corps d'une requête est trop grand.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

Codes de statut HTTP renvoyé avec l'exception.

- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande qui est trop grande.

ReadOnlyViolationException (structure)

Générée lorsqu'une demande tente d'écrire des données dans une ressource en lecture seule.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande dans laquelle il manque le paramètre.

S3Exception (structure)

Générée en cas de problème lors de l'accès à Amazon S3.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

ID de la demande en question.

ServerShutdownException (structure)

Générée lorsque le serveur s'arrête pendant le traitement d'une demande.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

StatisticsNotAvailableException (structure)

Générée lorsque les statistiques nécessaires pour satisfaire une demande ne sont pas disponibles.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

StreamRecordsNotFoundException (structure)

Générée lorsque les enregistrements de flux demandés par une requête sont introuvables.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

ThrottlingException (structure)

Générée lorsque la fréquence des demandes dépasse le débit maximum. Les demandes peuvent faire l'objet d'une nouvelle tentative une fois cette exception genérée.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande qui n'a pas pu être traitée pour cette raison.

TimeLimitExceededException (structure)

Générée lorsque l'opération dépasse le délai imparti pour celle-ci.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

Message détaillé décrivant le problème.

- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande qui n'a pas pu être traitée pour cette raison.

TooManyRequestsException (structure)

Générée lorsque le nombre de demandes traitées dépasse la limite.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande qui n'a pas pu être traitée pour cette raison.

UnsupportedOperationException (structure)

Générée lorsqu'une demande tente de lancer une opération qui n'est pas prise en charge.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

UnloadUrlAccessDeniedException (structure)

Générée lorsque l'accès est refusé à une URL qui est une cible de déchargement.

Champs

- code : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Codes de statut HTTP renvoyé avec l'exception.
- detailedMessage : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  Message détaillé décrivant le problème.
- requestId : obligatoire : chaîne de type : `string` (chaîne encodée en UTF-8).

  ID de la demande en question.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.